Il Mainframe

Una grande tecnologia nel rispetto dell'ambiente

TERZA EDIZIONE

Ottobre 2010

Il Mainframe

Una grande tecnologia nel rispetto dell'ambiente

Indice

Indice 2
Gli autori 6
Ringraziamenti 9
Premessa 10
1.0 Introduzione ai Sistemi Centrali 12
1.0.0 Definizioni 13
1.0.1 Modelli Applicativi 15
1.0.2 Tipi di Lavoro (Workload) 20
1.0.3 Compiti e Ruoli 21
1.1 Architettura dei Sistemi Mainframe 23
1.1.0 Modelli Architetturali 23
1.1.1 La z/Architecture 25
1.1.2 Architettura di I/O 38
2.0 Hardware dei Sistemi Centrali IBM (z10EC) 46
2.0.0 PU Chip z10 46
2.0.1 SC-SMP Hub Chip z10 51
2.0.2 z10 Multiple Chip Module (MCM) 52
2.0.3 z10 Book 54
2.0.4 La specializzazione dei processori 57
2.0.5 Il Channel Subsystem z10 60
2.0.6 Gli Adapter di I/O z10 64
2.0.7 Support Element e HW Management Console 67
2.0.8 High Performance Ficon 68
2.0.9 Fibre Channel Protocol per SCSI 70
2.1 Hardware di IBM zEnterprise 72
2.2 La Virtualizzazione 77
2.2.0 Concetti Generali 77
2.2.1 Caratteristiche della Virtualizzazione 78
2.2.2 Le ragioni della Virtualizzazione 78
2.2.4 Le risorse virtualizzabili 80
2.3 Virtualizzazione HW e SW 80
2.3.0 Le diverse opzioni disponibili 81
2.3.1 Il Partition Resource & System Management (PR/SM) 82
2.3.2 PR/SM - Caratteristiche delle partizioni logiche 84
2.3.3 PR/SM - Gestione I/O - Multi Image Facility (MIF) 85
2.3.4 Risorse di I/O virtuali (HiperSocket) 86
2.3.5 La potenza dei Mainframe virtualizzati 86
3.0 Il Sistema Operativo z/OS 91
3.0.0 Introduzione 91
3.0.1 L'Address Space 93
3.0.2 La gestione della Memoria Reale 100
3.0.3 Interfacce Utente (User Interfaces) 104
3.0.4 Il JES2 (Job Entry Subsystem) 111
3.0.5 Job Control Language (JCL) 115

3.0.6 Spool Display & Search Facility (SDSF) 118
3.0.7 Il Workload Manager (WLM) 122
3.0.8 Gestione dei dati in z/OS 124
3.0.9 Gestori di basi dati (DBMS) 134
3.0.10 Disegnare Applicazioni per z/OS 142
3.0.11 Linguaggi di Programmazione 144
3.0.12 Sottosistemi Transazionali 146
3.0.13 Soluzioni Applicative di Business Intelligence (BI) 153
3.0.13.1 Introduzione 153
3.0.13.2 Alcune definizioni 153
3.0.13.3 La Business intelligence nei sistemi z 155
3.0.13.4 Perché realizzare un Data warehouse in ambiente mainframe 157
3.0.14 z/OS Management Facility 159
3.0.15 Hiperdispatch 162
3.0.16 Il cluster di z/OS (Parallel Sysplex) 167
3.0.17 UNIX System Services 169
3.1 La Virtualizzazione Software z/VM 173
3.1.0 Il Sistema Operativo z/VM - Caratteristiche 173
3.1.1 Le Macchine Virtuali 174
3.1.2 Componenti di Base 176
3.1.2.1 Control Program (CP) 176
3.1.2.2 Il CMS 180
3.1.3 Dispositivi di I/O in ambiente z/VM 186
3.1.3.1 Unità a Disco 186
3.1.3.2 Unità a Nastro Virtuale 187
3.1.3.3 Spool devices 187
3.1.4 Gestione dispositivi di rete 188
3.1.4.1 z/VM Guest LAN 188
3.1.4.2 z/VM Virtual Switch 190
3.1.5 Virtualizzazione di CPU e Memoria 192
3.1.6 Scheduling e Dispatching 194
3.1.7 Start Interpretive Execution (SIE) 196
3.2 Linux nei Sistemi Centrali IBM 199
3.2.0 Utilizzi di Linux in ambienti enterprise 199
3.2.1 Le applicazioni su z/Linux e il processo di porting 200
3.2.2 Linux sul Mainframe – Modi Operativi e Risorse Gestite 201
3.2.3 Uso di processori specializzati (IFL) 203
3.2.4 Networking 204
3.2.5 Linux nei Sistemi Virtualizzati con z/VM 205
3.2.6 Distribuzioni 207
3.2.7 Il processo di installazione 207
3.2.8 La clonazione di Sistemi Linux sotto z/VM 208
3.3 Il Sistema Operativo z/VSE 209
3.3.0 Storia dello z/VSE 209
3.3.1 Architettura 211
3.3.2 I prodotti 213
3.3.3 I connettori (z/VSE e-business connectors) 214
3.3.4 La sicurezza in ambiente z/VSE 221
3.3.5 Scenari di comunicazione e utilizzo 224

3.3.6 Previsioni di sviluppo per le prossime versioni 226
4.0 La selezione della Piattaforma Informatica e l'Ottimizzazione della Infrastruttura 228
4.0.0 Introduzione 228
4.0.1 Definizioni 229
4.0.2 Portabilità, Standard, Openness 236
4.0.3 Requisiti Non Funzionali 241
4.0.4 I driver della Platform selection 243
4.0.5 Il Metodo 246
4.0.6 Il Valore della Tecnologia 248
4.0.7 Confronto tra Sistemi Eterogenei 251
4.0.8 Calcolo delle componenti di Costo [TCO] 262
4.0.9 Un Esempio: Scelta della Piattaforma 268
4.0.10 Un Esempio: Cambio di piattaforma 273
4.0.11 Ottimizzazione della Infrastruttura Informatica 277
4.0.12 Fattori che influenzano il processo di ITRO 279
4.0.13 Il Consolidamento dei Serventi 280
4.0.14 Un Esempio: Valutazione di un processo di Server Consolidation 284
4.0.15 Conclusioni 289
5.0 La famiglia dei Mainframe 290
Appendice A: I sistemi ibridi - una possibile evoluzione dei sistemi mainframe 295
Appendice B: Sicurezza e Crittografia dei Mainframe 312
Appendice C: I sottosistemi Storage 332
Elenco delle figure 350
Bibliografia – Pubblicazioni - Link 356

Gli autori

Angelo Barbarino Senior Consulting IT Specialist del Systems & Technology Group, IBM Italia. Dal 2005 è impegnato a favorire la collaborazione didattica con le università italiane, promuovendo il corso "Sistemi Centrali" del quale ha tenuto molte edizioni in diversi atenei. Contribuisce alla realizzazione di progetti di ottimizzazione di infrastrutture informatiche a livello europeo. Laureato in Fisica a Catania nel 1981, segue da circa trenta anni il settore informatico. È membro del TEC - Technical Expert Council di IBM Italia e partecipa a diverse comunità e gruppi di lavoro a livello internazionale.

Francesco Bertagnolli IT Specialist del Systems & Technology Group, IBM Italia, si è laureato in Ingegneria Elettronica presso l'Università degli Studi di Pavia. Dopo aver lavorato come dottorando al Laboratorio di TLC dell'Università di Pavia, nel 2007 entra in IBM dove consegue un master in Business Administration al Politecnico di Milano. Inizialmente si è occupato di architetture multicore e del processore Cell/BE e dal 2009 ha responsabilità tecniche sui sistemi z, prevalentemente in ambito z/OS. In qualità di IBM University Ambassador contribuisce a diffondere le conoscenze dell'architettura dei sistemi centrali presso gli atenei italiani.

Andrea Corona Advisory IT Specialist del Systems & Technology Group, IBM Italia. Laureato in Ingegneria Elettronica nel 2002 presso l'Università di Cagliari, è in IBM dal 2003. Contribuisce alla realizzazione di progetti complessi per clienti di diversi settori eterogenei. Partecipa a team di ricerca sulle performance e sulla sicurezza delle applicazioni Web J2EE e Linux. Ha assunto dal 2006 al 2008 responsabilità tecnico-commerciali sui sistemi z, con focus sull'introduzione di nuove tecnologie in questa piattaforma. Nel 2008 ha lavorato per il Mainframe Benchmark Center di Montpellier. Nel 2009 è stato team-leader di un progetto di ricerca di High Performance Computing in campo medico. Dal 2010 si occupa di progetti innovativi di Cloud Computing e Smarter Healthcare. E' coautore della pubblicazione a cura di IBM "The IBM Mainframe Today: System z Strengths and Values**"** e collabora alla IBM Academic Initiative.

Marco De Felice Laureato in Fisica con indirizzo informatico presso l'Università di Genova, ha lavorato per quasi 20 anni in IBM occupandosi sempre di mainframe. Uscito dall'IBM del 1996, ha continuato a lavorare come consulente, agente e rivenditore in società partner IBM. Nel 2001 ha costituito con altri soci una nuova società di cui è responsabile, **Delphis Informatica** srl di Genova, Business Partner IBM, che ha competeze e certificazioni avanzate nelle aree Linux, virtualizzazione e sistemi centrali, disponendo tra l'altro di due sistemi mainframe per prove, test e services. La Delphis Informatica offre consulenza e sviluppa progetti, prevalentemente nell'area z/VSE, z/VM, zLinux, DB2 e tools, per attività di tuning e studi di architettura sui sistemi centrali.

Marco Gardini IT Operations Manager di **Fratelli Carli** ha ricoperto il ruolo di Mainframe System Architect dal 1990 con la responsabilità di coordinare tutti gli aspetti infrastrutturali del sistema informativo, con particolare riferimento ai sistemi operativi z/VM, zLinux e z/VSE. La competenza e l'esperienza maturata in queste aree gli hanno permesso di realizzare soluzioni tecniche e applicative all'avanguardia, soprattutto per quanto riguarda l'interconnessione fra i vari ambienti, utilizzando al meglio le potenzialità di Linux su piattaforma mainframe. Ha frequentato innumerevoli corsi di specializzazione in Italia e all'estero, partecipando, spesso anche in qualità di relatore, a importanti eventi internazionali quali il GSE (Guide Share Europe) e il WAVV (World Alliance VM VSE) ottenendo prestigiose certificazioni.

Gaetano Maretto Laureato in Ingegneria Chimica nel 1968 è entrato in IBM nel 1970. Ha seguito l'evoluzione della piattaforma mainframe durante tutta la sua carriera. Nel suo ruolo di IT Specialist è stato coinvolto nell'introduzione di nuove importanti tecnologie sulla piattaforma mainframe, quali la connessione in fibra ottica (ESCON, FICON), il clustering (Parallel Sysplex), la gestione autonomica del carico di lavoro (Workload Management). Ha sviluppato corsi che sono stati poi usati per addestramento sia interno che verso i clienti. Ha collaborato con istituzioni universitarie con seminari di approfondimento su temi quali la virtualizzazione, le tecnologie ottiche e l'architettura dei mainframe. In pensione da febbraio 2010, continua a collaborare con IBM, contribuendo alla redazione di articoli e pubblicazioni relative alla piattaforma.

Mario Moretti Laureato in Matematica presso la Sapienza Università di Roma, ha maturato una trentennale esperienza tecnica in area mainframe con varie mansioni. In particolare dal 1983 ha svolto la sua attività al Laboratorio di Sviluppo Software IBM Tivoli di Roma. Attualmente è IT Specialist del Systems & Technology Group, IBM Italia. Partecipa alla IBM Academic Initiative a supporto delle conoscenze e collaborazioni con l'ambiente universitario. Dal 2007 tiene il corso "Sistemi Centrali" alla Sapienza Università di Roma.

Filippo Quarta Laureato in Ingegneria Informatica nel 2001 presso l'Università di Lecce, è referente tecnico per i nuovi workload sui sistemi z del Systems & Technology Group, IBM Italia. E' in IBM dal 2007 e svolge attività tecnico-commerciali in ambiente mainframe e zLinux. E' Novell Linux Certified Professional e Data Center Technical Specialist dal 2009. In precedenza ha ricoperto il ruolo di ricercatore industriale in Telecom Italia contribuendo allo sviluppo di diverse piattaforme per servizi mobili e partecipando a progetti di ricerca legati alla convergenza delle reti.

Corrado Troncone Dopo aver concluso il biennio all'Università Federico II di Napoli, Facoltà d'Ingegneria, entra in IBM nel 1983 come Tecnico di Manutenzione Software sui Sistemi Centrali di architettura IBM/370, sistemi operativi MVS e VM. Nel 1987 viene assegnato al Laboratorio di Sviluppo IBM VM di Kingston, NY (USA); tornato in Italia svolge attività sistemistica sui mainframe. Dal 2000 fa parte del Brand Technical Support group continuando a occuparsi di sistemi z nel Systems & Technology Group, IBM Italia. Partecipa alla IBM Academic Initiative per la diffusione delle conoscenze dell'architettura dei Sistemi Centrali presso le Università italiane.

Ringraziamenti

Oltre agli **autori**, ringraziamo i colleghi della IBM Italia per il loro prezioso contributo:

Giorgio Acquafresca, Client Technical Specialist, **Massimiliano Castellini**, Executive IT Specialist, **Massimo Leoni**, Executive IT Architect, **Marita Prassolo**, Distinguished Engineer, **Stefano Ricci**, Client Technical Architect, **Francesco Silveri**, Director of STG Technical Sales.

Premessa

[a cura di Francesco Silveri]

La terza edizione del volume "Il Mainframe" è motivata dal grande successo delle precedenti edizioni (la prima nel 2007 e la seconda nel 2008) con più di duemila copie distribuite all'interno e all'esterno di IBM, dal continuo lavoro di aggiornamento, ampliamento e sviluppo in linea con l'evoluzione tecnologica, e dall'esigenza di soddisfare un pubblico sempre più numeroso e diversificato.

Per coloro che già conoscono questa pubblicazione, voglio indicare subito le principali novità:

- *maggiori informazioni sulle tipologie di workload e i criteri di selezione della piattaforma;*
- *un nuovo capitolo dedicato al sistema operativo z/VSE;*
- *un'ampliamento dell'appendice sui sistemi ibridi, con alcune informazioni sul nuovo IBM zEnterprise System;*
- *una appendice sullo storage.*

Per chi legge questo libro per la prima volta, voglio ricordare che questo testo nasce dal lavoro di un gruppo di colleghi IBM che alcuni anni fa hanno iniziato un'esperienza di insegnamento nelle Università italiane sulla tecnologia mainframe. Dai loro appunti è nato il primo prototipo che, con la collaborazione di altre persone, è diventato un libro. Si tratta di un documento "made in Italy" a supporto dei cicli di lezioni universitarie che progressivamente è diventato il "manuale del mainframe", una guida rapida e completa per capire i fondamenti prima di approfondire le conoscenze nei contenuti e nella pratica.

Il lettore, seguendo i capitoli di questo libro, può scoprire come IBM considera il mainframe la piattaforma di innovazione e di eccellenza. Il mainframe ha quindi un ruolo di assoluto valore nel panorama dell'informatica moderna e, pur mantenendo un forte grado di compatibilità con le applicazioni esistenti, si propone come la piattaforma più adatta a soddisfare le sempre più complesse esigenze applicative e di sistema del prossimo futuro.

Oggi i Mainframe sono presenti - ed hanno un ruolo insostituibile in tutto il mondo - nelle infrastrutture informatiche delle più importanti aziende industriali e finanziarie, nelle società di servizi pubbliche e private e nelle grandi istituzioni nazionali ed internazionali. Il termine stesso "Mainframe" è diventato sinonimo di server di grandi capacità elaborative ed alto livello di qualità del servizio. Fare di questa tecnologia materia di studio nelle aule Universitarie è un progetto di grande valore e di vasta portata che offre agli studenti un programma di fondamenti avanzati di informatica, con l'importante obiettivo di trasferire concetti e tecniche di grande valore formativo.

Naturalmente la lettura di questo testo risulta utile anche a chi già si occupa di informatica e voglia conoscere l'architettura e le caratteristiche del mainframe. Per questa ragione recentemente si sono aggiunti nuovi lettori che già operano in varie aree di questo settore.

Ho avuto l'onore di essere tra i primi lettori di questo libro nel 2007. Ricordo di averlo letto tutto di seguito per cogliere l'opportunità di riorganizzare in modo ordinato e preciso le conoscenze che avevo di questa piattaforma. Ho apprezzato il lavoro che gli autori, ed amici, avevano fatto con passione e con grande esperienza didattica (quest'ultima mi aveva davvero piacevolmente sorpreso).

Ora ho il grande piacere di scrivere le prime righe della terza edizione con l'augurio a chi seguirà i corsi universitari e a chi leggerà questo volume di poter sfruttare al meglio nella propria vita professionale le conoscenze apprese.

Conoscendo i piani di evoluzione del mainframe IBM vi posso garantire che questo libro avrà nuove edizioni!

1.0 Introduzione ai Sistemi Centrali

Da più di quarant'anni una parte significativa dei processi produttivi e dei servizi viene supportata da una infrastruttura informatica costituita dai Sistemi Centrali. Fino a metà degli anni '70 la storia stessa dell'informatica e dei sistemi operativi si identificava con tali sistemi anche detti "Mainframe". La maggior parte delle istituzioni private o pubbliche, nei più disparati settori della economia o dei servizi, basano la parte più delicata e strategica della loro attività su computer di questo tipo.

I Sistemi Centrali sono frutto di ingenti investimenti da parte di case produttrici e oggi attraversano una significativa fase di innovazione proiettata verso un loro inserimento in scenari eterogenei ed "aperti" per tecnologie e caratteristiche.

Risulta singolare che esista poca informazione su testi divulgativi ed in internet su tali argomenti ad eccezione di quella espressa dalle case produttrici.

Obiettivo di questo testo è di fornire un contributo alla descrizione di alcuni aspetti di questa materia.

1.0.0 Definizioni

Sistema Centrale

Si definisce Sistema Centrale un calcolatore dotato di elevata capacità elaborativa usato per gestire grandi volumi di dati con un elevato livello di sicurezza ed affidabilità. Tale sistema è acceduto da un alto numero di utenti contemporaneamente. Esso gestisce un carico di lavoro misto eseguendo attività tra loro differenti per caratteristiche e impegno elaborativo.

Figura 1 Sistema centrale

Queste attività vengono eseguite in contemporanea mantenendo priorità assegnate ed armonizzando le varie necessità elaborative. Il Sistema Centrale si contrappone a server monouso, in grado cioè di svolgere un solo tipo di workload (firewall, router, file server, print server, ecc.).
Lo scenario in cui il sistema centrale si inserisce è cambiato nel tempo. Inizialmente il sistema centrale accentrava su di sé tutte le funzioni informatiche (controllo degli accessi, elaborazioni con calcoli numerici complessi – ad esempio scientifiche – gestione delle reti ed altro). Oggi il sistema centrale è collocato in una configurazione di server in cui una serie di funzioni sono demandate a server monouso poiché questa infrastruttura è quella più efficiente dal punto di vista dei costi. Possiamo dire che nei sistemi centrali sono rimaste tre funzioni fondamentali: Data base server, Application server e Presentation server. Questi concetti verranno sviluppati in seguito. Questa cooperazione è stata resa possibile poiché i sistemi centrali hanno implementato tutti gli standard adottati e richiesti dal mercato IT, siano essi de jure o de facto (TCP/IP, Linux, POSIX, Java, XML ecc.); in questo modo l'integrazione dei sistemi centrali all'interno di una rete di elaboratori eterogenea risulta facile e completa.

Il Sistema Centrale per eccellenza e per anzianità di servizio è il Mainframe. La sua storia verrà descritta più avanti. Basti dire che oggi il termine Mainframe è sinonimo di elaborazione sicura, affidabile, autonomica e flessibile. A partire da metà degli anni '90 diversi produttori di computer hanno introdotto dei modelli di sistemi centrali esaltandone la maggiore o minore affinità con le caratteristiche del Mainframe, sebbene utilizzino tecnologie e architetture diverse.

Infrastruttura Informatica

Si definisce Infrastruttura Informatica l'insieme di hardware, software e collegamenti che implementano lo scenario in cui categorie differenti di utenti accedono a risorse elaborative quali dati e programmi. Spesso il disegno di una infrastruttura informatica rappresenta un processo continuo di adattamento fra le esigenze del presente e gli sviluppi futuri che si possono prevedere. Tuttavia alcuni paradigmi, quali flessibilità ed affidabilità della struttura informatica, costituiscono punti fermi del progetto di una infrastruttura informatica: quest'ultima non deve mai rappresentare un ostacolo alla dinamicità dei processi di business che creano il vantaggio competitivo tra le aziende.

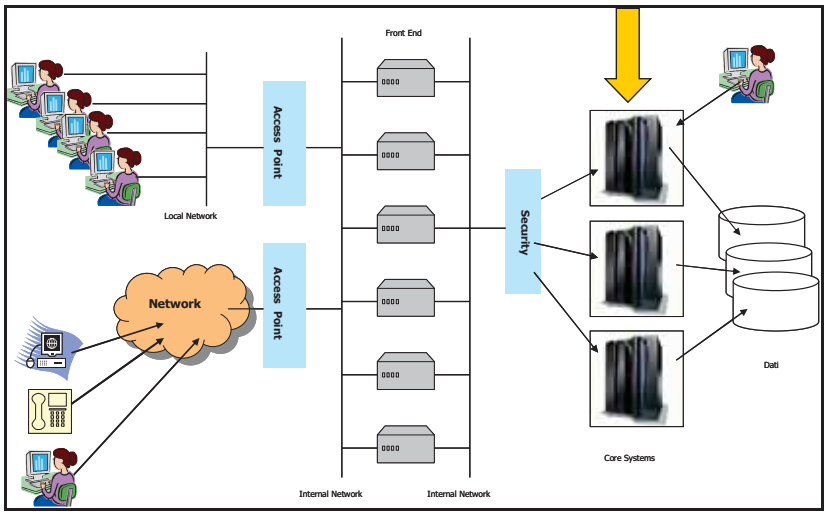


Figura 2 Infrastruttura informatica

Piattaforma informatica

Si definisce Piattaforma informatica l'insieme di un calcolatore e di un sistema operativo. Tale termine consente di definire uno scenario elaborativo identificando le caratteristiche salienti dell'hardware e del software che su di esso opera.

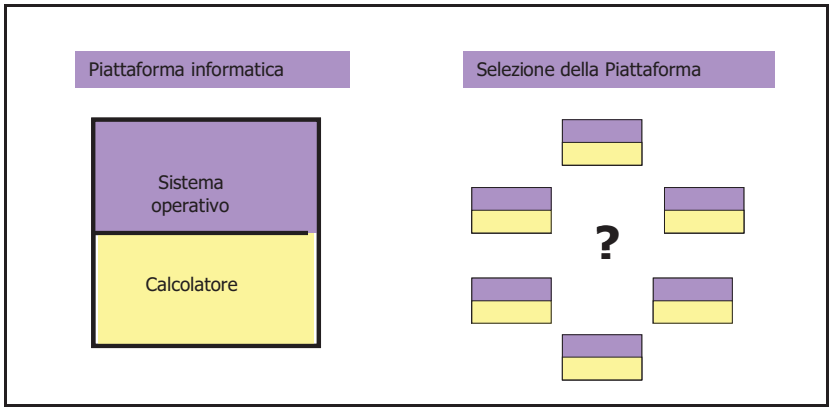


Figura 3 Piattaforma informatica

A tal riguardo esiste un processo di valutazione di una o più piattaforme all'interno di una infrastruttura informatica. Questo processo chiamato "selezione della piattaforma" si basa sull'insieme dei requisiti richiesti per una certa infrastruttura e sulle caratteristiche dei prodotti in grado di rispondere a tali esigenze.

Al compito tecnologico (Selezione della Piattaforma) si abbina la valutazione del costo di acquisizione di tali prodotti, detto TCA (Total Cost of Acqusition), e del costo di gestione, detto TCO (Total Cost of Ownership) che comprendono l'esercizio e la manutenzione dell'attrezzatura. Questi argomenti saranno ripresi con maggiore dettaglio in un capitolo successivo.

1.0.1 Modelli Applicativi

In questo paragrafo introduciamo i modelli infrastrutturali che maggiormente hanno caratterizzato l'evoluzione dei sistemi centrali negli anni recenti. Questi cambiamenti sono dovuti alle trasformazioni subite dalle organizzazioni aziendali e al rapporto fra queste e il mercato in cui operano.
A partire dalla fine degli anni '90, con l'esplosione del fenomeno internet e dei mercati globali, è emerso un nuovo disegno di azienda, non più monolitica e chiusa, ma "aperta" e con interazioni sempre più strette con agenti esterni (fornitori, clienti, ecc.), flessibilità sempre più spinte e in grado di riposizionare costantemente la propria attività. Tutto ciò, supportato dalle nuove tecnologie, ha influito sui modelli infrastrutturali.

I modelli infrastrutturali sono quattro:

- Modello Host Centrico
- Modello Client/Server
- Modello Web Application
- Modello Service Oriented

Modello Host Centrico

Il modello host centrico è legato alla visione del sistema centrale come "hub" di applicazioni e dati all'interno del quale vengono eseguite le principali attività in completa autonomia e con scarse interazioni dei processi con enti esterni. E' un sistema intrinsecamente chiuso che molto spesso utilizza interfacce e applicazioni fatte in casa e ha una limitata adesione agli standard informatici.
Esso ha una flessibilità molto ridotta poiché ogni innovazione e modifica deve essere costruita "ad hoc". Gli utenti che si connettono al sistema sono già ben identificati (login e password) ed in genere sono dipendenti dell'azienda; questa circostanza garantisce una buona sicurezza informatica. La interazione con internet è molto bassa e in genere non legata alla cooperazione di processi aziendali. La gestione del sistema è affidata in genere a tecnici con specializzazione consolidata sui sistemi centrali ma limitata nelle nuove tecnologie.
In questo paradigma l'utente finale dispone di terminali con nessuna funzione evoluta se non la visualizzazione di caratteri (anche a colori).
Questa limitazione è imposta anche dalla scarsa potenza delle reti trasmissive e dai suoi alti costi.

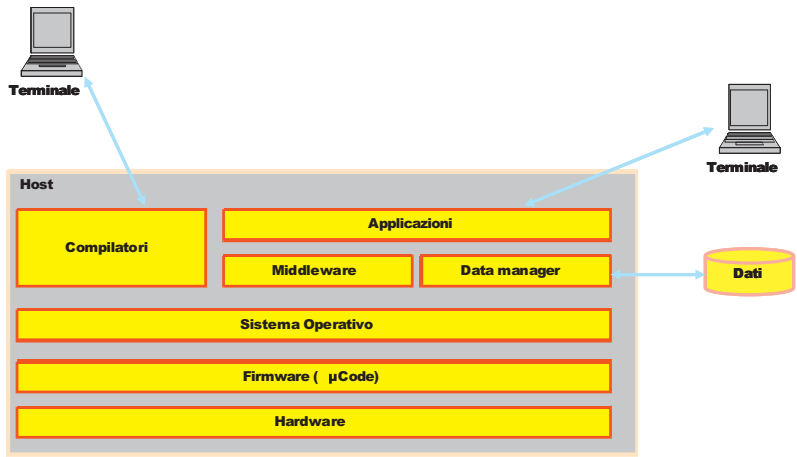


Figura 4 Modello applicativo Host Centrico

Modello Client/Server

Dalla fine degli anni '80 si è assisitito ad una evoluzione molto rapida delle reti di telecomunicazione: banda passante sempre più estesa, costi di connessione sempre più bassi, l'affermazione dello standard TCP/IP e la disponibilità di potenze elaborative individuali (il personal computer). Perciò un nuovo modello infrastrutturale ha preso forma: il modello Client/Server. In accordo con questo modello, un sistema monolitico

viene progressivamente suddiviso in processi client (cioè richiedenti) e processi server (cioè fornitori di servizio) allocati su unità elaborative logicamente (e spesso anche fisicamente) distinte. L'elemento centrale del sistema è la rete. Tipici server sono i server dati (DBMS), i server di applicazioni, i server di sviluppo, dove ad esempio vengono eseguiti i compilatori; l'utenza, non è più necessariamente preidentificata e utilizza interfacce grafiche (GUI) per accedere alle applicazioni.

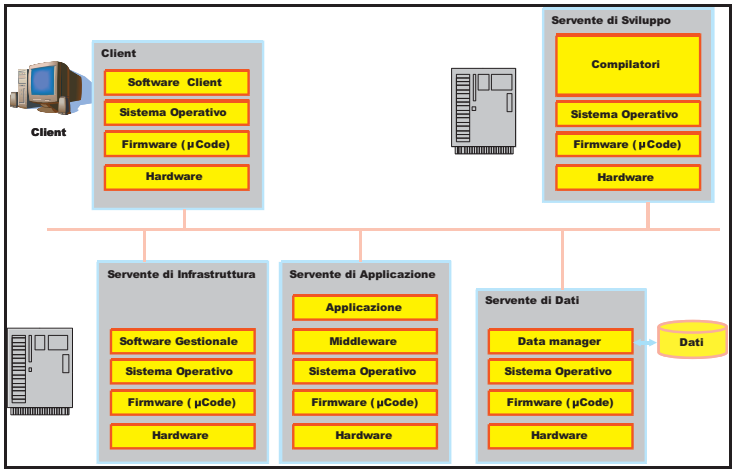


Figura 5 Modello applicativo Client/Server

In sintesi su una rete (locale o meno) si attestano diversi Personal Computer dotati di GUI, che nell'insieme fanno una parte o tutto ciò che in precedenza veniva svolto da un grande calcolatore mainframe. I programmi client e server presentano delle "rigidità", ad esempio nella non standardizzazione dell'interfaccia GUI, ma il più importante svantaggio di questa soluzione è nel non efficiente utilizzo delle risorse disponibili. Ad esempio se un server è carico di lavoro (ad esempio per la gestione dei dati) rispetto ad un altro che è libero, quest'ultimo non può mettere le proprie risorse a disposizione del primo server generando significativi colli di bottiglia.

Modello WEB Application

Un passo avanti rispetto al modello Client/Server è rappresentato dal modello Web Application. In questo modello il client utilizza in genere come interfaccia un programma "internet browser" di internet (o un applet Java) ed i servizi di connettività applicativa sono forniti da un programma di tipo "web server". In pratica si standardizza l'accesso alle applicazioni in maniera sempre più integrata con Internet e gli standard collegati. I protocolli di comunicazione TCP/IP e http sono alla base di questo modello.

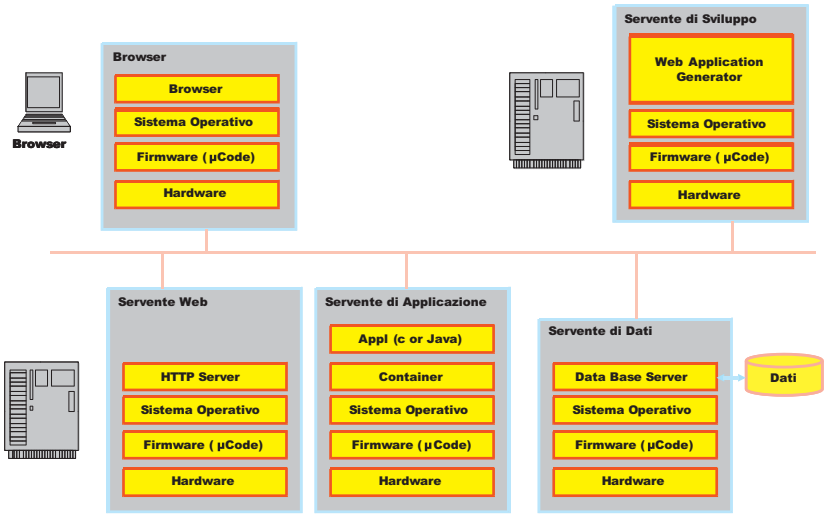


Figura 6 Modello applicativo Web Application

Nel modello, un importante ruolo è rivestito dai "contenitori" applicativi (Web Application Server) in grado di ospitare applicazioni scritte con linguaggi di programmazione ad oggetti (ad es. Java).

Modello Service Oriented

Un ulteriore passo avanti nel campo della flessibilità ed apertura è rappresentato dal modello SOA (Service Oriented Architecture), il cui scopo ultimo è il connettere una grande varietà di sistemi senza vincoli legati a software proprietario raggiungendo una vera interoperabilità.
Secondo questo modello è possibile far cooperare strettamente due programmi scritti in linguaggi differenti che sono eseguiti su sistemi operativi differenti senza l'intervento di una qualche parte di codice proprietario e a costi accettabili.

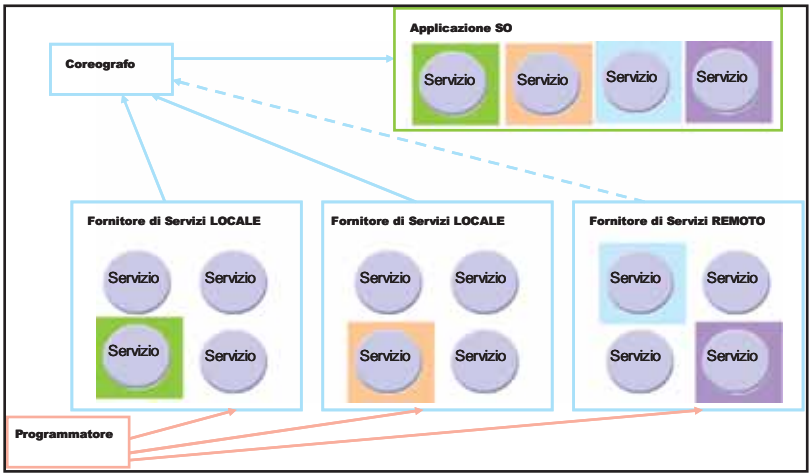


Figura 7 Coreografia SOA

Questa interoperabilità è oggi ancor più raggiungibile se la connessione operativa è realizzata attraverso il linguaggio di interscambio XML (Extended Markup Language). Un'architettura a servizi, la SOA appunto, utilizza protocolli e linguaggi operativi standard (SOAP UDDI, WSDL, XML) e consente di realizzare un'applicazione complessa formata da servizi Web, indifferenti alla piattaforma informatica che li ospita, coordinati da un programma di "coreografia".

In altre parole questo modello gestito dal SOA Server (Server di Servizi) è un nuovo modo di implementare le applicazioni, viste come un insieme di servizi coordinati che una volta combinati offrono un servizio finale più complesso. Un vantaggio molto rilevante del modello a servizi è rappresentato dalla maggiore riusabilità dei componenti (Web Service) che rendono più veloce ed economica la realizzazione di nuove applicazioni.

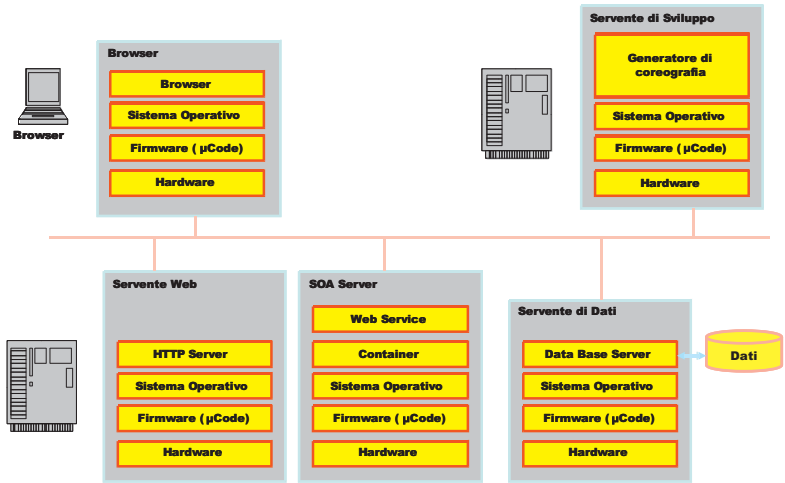


Figura 8 Modello applicativo SOA

Un esempio di utilizzo dell'architettura service oriented può essere un ente pubblico che espone sul web i risultati di applicazioni COBOL (software legacy). Le stesse applicazioni COBOL, opportunamente trattate, potrebbero essere rese disponibili alle applicazioni di altri enti mediante l'utilizzo delle interfacce standard SOA.

Il modello applicativo prevede l'esistenza di un contenitore logico di questi Web Service chiamato SOA Server dotato di tutti i requisiti in termini di standard applicativi e di connessione. Esiste anche un generatore della coreografia che in ogni momento gestisce su server locali o remoti (allocati su piattaforme informatiche potenzialmente diverse) la sequenza dei processi applicativi necessaria per ottenere un dato risultato finale.

I Sistemi Centrali sono in grado di ospitare i quattro modelli applicativi descritti ovvero possono operare come un sistema Host Centrico, possono operare da Server di dati ed Applicazioni o entrambi, possono fungere da Web server o Application o Data Server per una Web Application e possono infine coreografare e fornire servizi (ad es. Web Services) in una Architettura Orientata ai Servizi (SOA).

1.0.2 Tipi di Lavoro (Workload)

I sistemi centrali gestiscono contemporaneamente Workload profondamente diversi fra di loro. In altre parole, una singola istanza di sistema operativo controlla centinaia o migliaia di processi aventi finalità e caratteristiche diverse. Tutto ciò è possibile in quanto tali sistemi sono multiprogrammati. Una suddivisione classica dei lavori è quella che distingue i cosidetti processi o job Batch (elaborazione a lotti) da quelli delle transazioni o online. I primi sono in genere caratterizzati dal fatto che una volta immessi nel sistema hanno una bassissima interazione con l'utente. Esempi di questa tipologia sono le stampe di fatture o bollette di utenze, oppure le riorganizzazioni dei

dati. La seconda tipologia di lavoro ovvero le transazioni online è invece caratterizzata da una vasta utenza che interagisce frequentemente con il sistema per richiedere e/o aggiornare dati contenuti in database attraverso programmi "ad hoc" chiamati

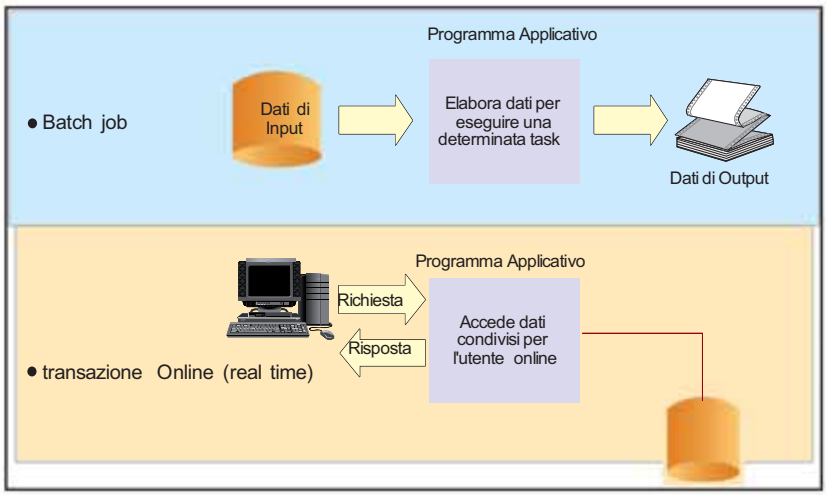


Figura 9 Tipologia di lavoro: Batch job e transazione online

Tutti questi lavori (batch e online) coesistono e usano le risorse del sistema centrale in base a politiche definite dagli specialisti che gestiscono il sistema. Queste politiche vengono poi elaborate da programmi speciali di sistema operativo, chiamati Schedulatori, che consentono di massimizzare il numero di processi per unità di tempo, garantendo i livelli di servizio definiti dai gestori del sistema.

1.0.3 Compiti e Ruoli

I sistemi centrali per la loro complessità ed importanza sono in genere gestiti da un pool di specialisti: la loro attività rende questi sistemi utili ed efficaci strumenti per il raggiungimento degli obiettivi di business. Gli investimenti in quest'area sono sempre cospicui e sono sempre confrontati con i guadagni di produttività e competitività.

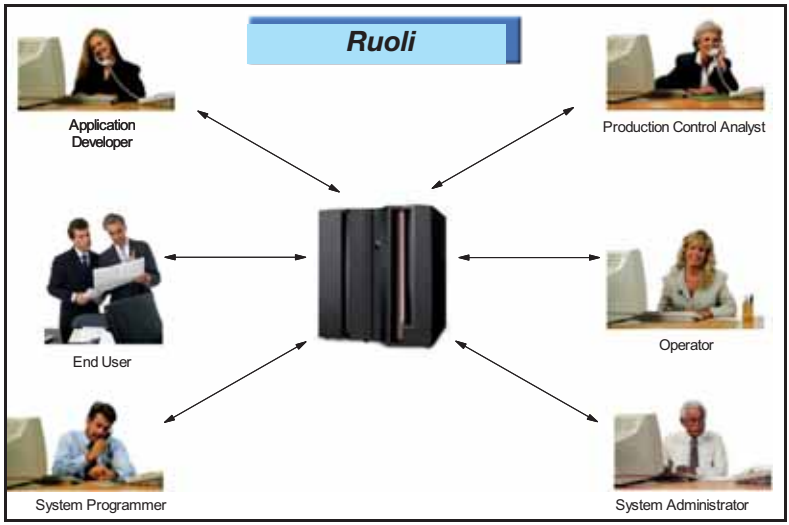


Figura 10 Compiti e Ruoli

Da questi punto di vista il personale dei sistemi informativi si può suddividere in:

Gestione e controllo del sistema

E' un gruppo di amministratori di sistema costituito da esperti di sistemi operativi, di gestione di dati, di gestione di reti, di gestione di programmi applicativi ed esperti di controllo della performance. A questi si possono aggiungere tutti coloro che si occupano del funzionamento operativo del sistema, cioè coloro che sono costantemente alla guida dei sistemi gestendo le console, rispondendo ai messaggi operativi, controllando il funzionamento dei dispositivi fisici operando in caso di errore e/o di ripartenze dei sistemi.

Sviluppo di nuove applicazioni

In questo reparto si trovano gli sviluppatori di applicazioni: coloro che analizzano le necessità applicative e disegnano le soluzioni, le implementano, le testano, le mettono in produzione e le mantengono.

Architetti di sistema

Sono le figure che disegnano le architetture informatiche, contattano consulenti indipendenti e/o esperti di aziende produttrici per valutare le possibili soluzioni in termini architetturali. In genere queste funzioni forniscono ai livelli decisionali tutte le informazioni che consentono gli investimenti in area informatica.

1.1 Architettura dei Sistemi Mainframe

In questo capitolo vedremo le caratteristiche principali dei Sistemi Centrali dal punto di vista del disegno architetturale, descriveremo la z/Architecture, cioè l'architettura dei mainframe IBM, in particolare verrà descritta l'organizzazione dei Sistemi, il relativo Instruction Set, la gestione della Memoria e dei dispositivi di Input/Output e la struttura dei sistemi Multiprocessori simmetrici. Verrà inoltre introdotto il concetto di virtualizzazione delle risorse HW; l'argomento sarà ripreso in modo più approfondito in un capitolo successivo, ma deve essere introdotto qui per stabilire una esatta terminologia nella descrizione achitetturale che segue.

1.1.0 Modelli Architetturali

Il modello architetturale secondo il quale sono progettati i Sistemi Centrali è basato sul modello della macchina di von Neumann. Gli elementi caratterizzanti di un calcolatore secondo von Neumann sono:

- memoria centrale, che contiene sia i dati sia il programma da eseguire ("stored program");
- CPU, ovvero il processore che elabora le istruzioni, utilizzando un'unità aritmetico logica (ALU) e una di controllo;
- dispositivi di I/O, utilizzati per inserire in memoria i dati d'ingresso di un programma e ricevere i risultati della elaborazione dalla memoria.

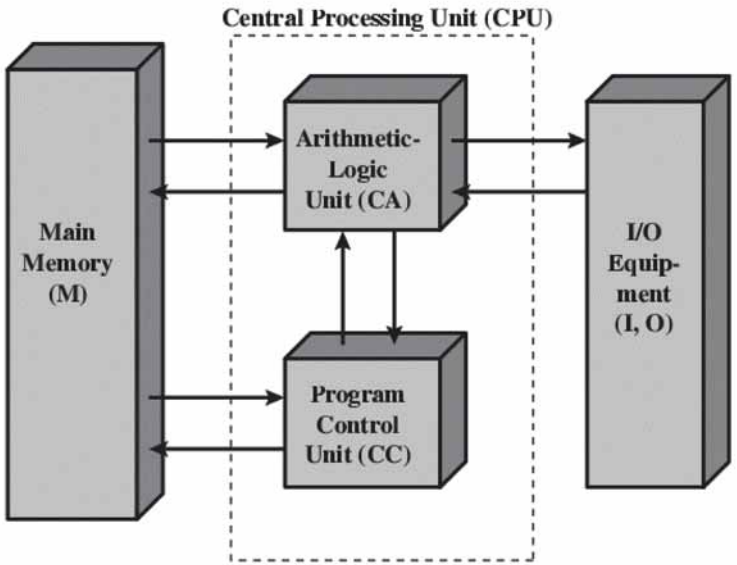


Figura 1 Struttura della macchina di von Neumann

I modelli architetturali dei calcolatori differiscono ad alto livello per il numero di elementi e la modalità di interconnessione fra di essi. Ad esempio, la struttura SMP (Symmetric Multi Processing), definita nel 1972, ha la caratteristica principale di utilizzare più di una CPU, con una sola memoria centrale ad accesso multiplo condivisa da tutti i processori. La z/Architecture dei mainframe è basata sulla struttura SMP. I processori e la memoria sono collegati da Bus ad alta velocità. Nelle implementazioni hardware solitamente si interpongono, per ragioni di efficienza, tra le CPU e la memoria centrale memorie più veloci, chiamate "cache". È importante notare che la struttura gerarchica della memoria è una caratteristica implementativa e non architetturale.

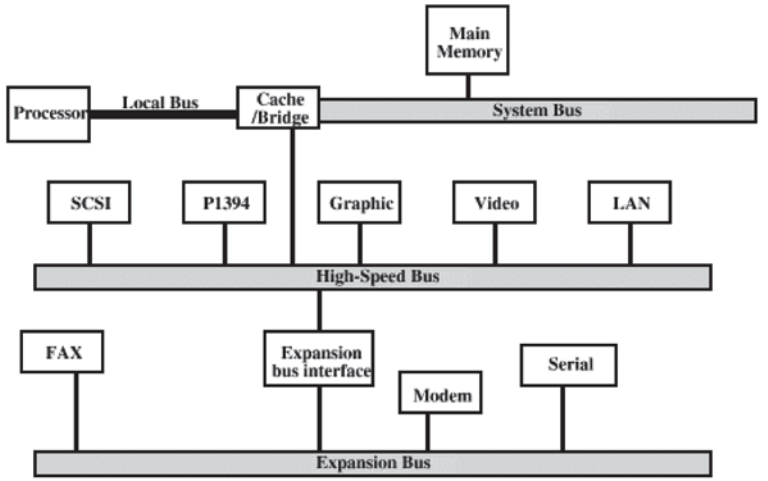


Figura 2 Schema di collegamento a Bus multiplo

Per quanto riguarda la comunicazione tra i dispositivi di I/O ("device"), le CPU e la memoria, solitamente si utilizzano i Bus, che sono dei collegamenti elettrici con circuiteria di controllo collegata. Per questioni di performance, specie se si collegano dispositivi che operano a velocità molto diverse, spesso non si utilizza un solo Bus ma più Bus (Struttura a Bus Multiplo – vedi figura 2). La z/Architecture presenta un sistema di interconnessione con i dispositivi più sofisticato del sistema a Bus multiplo, chiamato "Sistema a Canali". Questo tipo di sistema sarà descritto in seguito.

Infine definiamo Cluster di Sistemi un gruppo di calcolatori completi e interconnessi che operano come una risorsa elaborativa unificata. Ciascun componente del Cluster viene detto nodo. I Cluster di Mainframe sono gestiti mediante una complessa interazione fra Hardware, Sistema Operativo e Middleware. È possibile realizzare un Cluster di Sistemi Mainframe IBM detto Parallel Sysplex. Questo tipo di cluster sarà descritto in seguito.

1.1.1 La z/Architecture

Introduzione

L'architettura di un sistema ne definisce le caratteristiche dal punto di vista del software. Essa indica la struttura concettuale e il comportamento funzionale della macchina, che è tenuta distinta dalla realizzazione fisica della stessa. In particolare, l'architettura non si occupa dell'organizzazione dei flussi interni dei dati (data flow), del disegno fisico e delle prestazioni di una specifica implementazione della macchina; infatti, implementazioni di macchine diverse fra loro possono essere conformi ad una singola architettura. Se l'esecuzione di un insieme di programmi su implementazioni diverse produce i risultati che sono definiti da un'architettura, le implementazioni di questa sono considerate compatibili con tali programmi.

La z/Architecture è l'attuale architettura dei Sistemi Centrali ed è l'ultimo gradino nella scala evolutiva che parte dall'architettura System/360 (1964), cui seguono negli anni le architetture System/370 (1970), System/370 Extended Architecture (370-XA, 1983), Enterprise Systems Architecture/370 (ESA/370, 1988) e Enterprise Systems Architecture/390 (ESA/390, 1990).

La z/Architecture include tutte le funzioni dell'architettura ESA/390 ad eccezione di alcune funzioni utilizzate precedentemente solo dai sistemi operativi e non più necessarie. La z/Architecture fornisce inoltre estensioni significative all'architettura ESA/390, fra cui:

- l'estensione dei registri generali e di controllo a 64 bit;
- la modalità di indirizzamento della memoria a 64-bit, che si aggiunge a quelle a 24 e 31 bit presenti nell'architettura ESA/390; sia gli indirizzi degli operandi sia quelli delle istruzioni possono essere a 64 bit;
- l'espansione della Program-Status Word (PSW) a 16 bytes (128 bit) al fine di contenere gli indirizzi più lunghi;
- sino a tre livelli addizionali di tabelle per la traduzione dinamica degli indirizzi (Dynamic Address Translation - DAT), chiamate *Region Tables* e utilizzate per tradurre gli indirizzi virtuali a 64 bit.

Un punto fermo nella progettazione dell'architettura è stata la compatibilità binaria all'indietro garantita ai programmi scritti dagli utilizzatori del mainframe. Infatti, il criterio che guida le scelte dei progettisti hardware e software di Sistemi Centrali pone in grande considerazione la possibilità di salvaguardare il patrimonio applicativo degli utenti finali. Nell'introdurre nuove funzioni si garantisce la **compatibilità binaria** delle applicazioni - a livello di Instruction Set Architecture e di Application Binary Interface. Gli utenti finali hanno la libertà di scegliere se sviluppare nuove applicazioni avvantaggiandosi delle innovazioni tecnologiche, se continuare ad utilizzare le applicazioni con funzioni consolidate nel tempo o se sviluppare nuove applicazioni integrandole senza particolari problemi con quelle preesistenti (caso più frequente).

Il cuore della z/Architecture è descritto in dettaglio nel libro *z/Architecture Principles of Operation*[1], pubblicato da IBM la prima volta nel 2000 e tenuto costantemente aggiornato; esso costituisce, insieme ad altre pubblicazioni collegate, la documentazione di riferimento dell'architettura; descrive ogni funzione al livello di dettaglio necessario per scrivere un sistema operativo e preparare un programma in linguaggio Assembler. Le sue sezioni principali sono:

- **L'organizzazione del sistema**. Descrive le parti HW fondamentali del sistema (la memoria principale, la CPU, i sottosistemi di input/output, per citare i più importanti).
- **La memoria**. Spiega i differenti tipi di indirizzo (virtuale, reale, assoluto) le possibilità offerte dall'indirizzamento, le funzioni di protezione, la traduzione dinamica degli indirizzi (DAT); questa funzione, unita ad uno speciale supporto di programmazione, realizza la virtualizzazione della memoria.
- **Il controllo**. Descrive i servizi hardware per gestire lo stato del sistema; fra questi i più significativi sono: la misura del tempo, il coordinamento di una configurazione SMP, come avviare o fermare un sistema.
- **L'esecuzione dei programmi**. Spiega come le istruzioni HW che compongono qualunque programma vengono eseguite; descrive anche il formato delle istruzioni, vale a dire la loro rappresentazione binaria in memoria.
- **Le interruzioni**. Il meccanismo che permette alla CPU di cambiare il suo stato come effetto di eventi esterni o interni al sistema, sincroni o asincroni rispetto all'esecuzione dei programmi.
- **Le istruzioni.** Vengono descritte le istruzioni che formano l'Instruction Set dell'architettutra. Di ogni istruzione viene descritta l'esecuzione, la rappresentazione binaria, il suo funzionamento, gli operandi di cui ha bisogno e il modo in cui l'esecuzione può terminare; vi sono quattro grandi categorie di istruzioni: generali, decimali, in virgola mobile e di controllo del sistema.
- **Il Sottosistema di I/O.** E' la descrizione delle funzioni che permettono al SW in esecuzione sul Mainframe di scambiare dati con dispositivi esterni, quali adattatori di rete o sistemi storage. Questa sezione è una delle più ampie dell'intera architettura. E' bene ricordare che la z/Architecture **non** pone nessun vincolo all'implementazione di un dispositivo di I/O; un dispositivo di I/O sarà collegabile ad un mainframe se sarà in grado di scambiare dati secondo le funzioni e i protocolli descritti in questa sezione della z/Architecture.

L'organizzazione di un sistema Mainframe secondo la z/Architecture

Logicamente, un sistema organizzato secondo la z/Architecture è composto dai seguenti elementi HW:

- Memoria principale ("main storage" o "main memory")

[1] Il libro si può trovare alla pagina http://publibz.boulder.ibm.com/epubs/pdf/dz9zr006.pdf

- Una o più CPU
- Gli strumenti per la gestione (avvio/chiusura) del sistema (Operator Facilities)
- Un sottosistema a canali ("channel subsystem")
- I dispositivi di I/O.

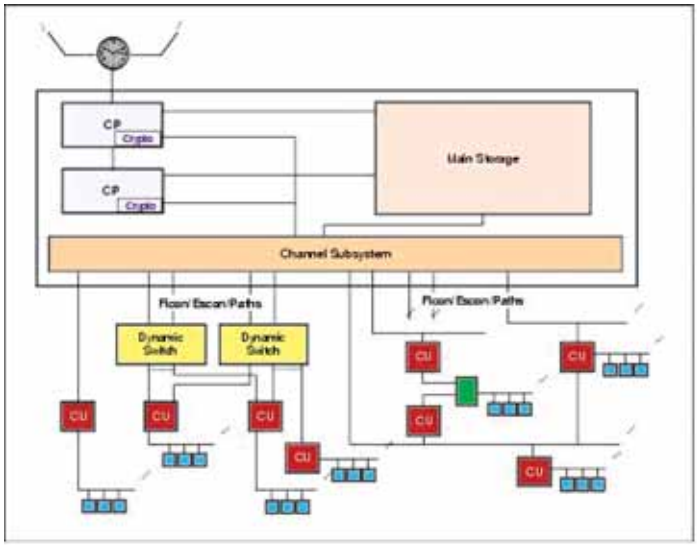


Figura 3 Organizzazione di un sistema mainframe secondo la z/Architecture

Da più di vent'anni è stata introdotta sulla piattaforma mainframe la virtualizzazione delle risorse HW allo scopo di aumentarne la flessibilità operativa. Di cosa si tratta? Si tratta di dividere le tre risorse HW (Potenza elaborativa, memoria e I/O) in parti (anche ineguali) e creare delle **partizioni logiche** a cui viene assegnata una porzione di ognuna delle tre risorse così divise. Un insieme di Potenza elaborativa, memoria e I/O così definito prende il nome di Partizione Logica (Logical partition – LPAR). La Figura 4 illustra questo concetto

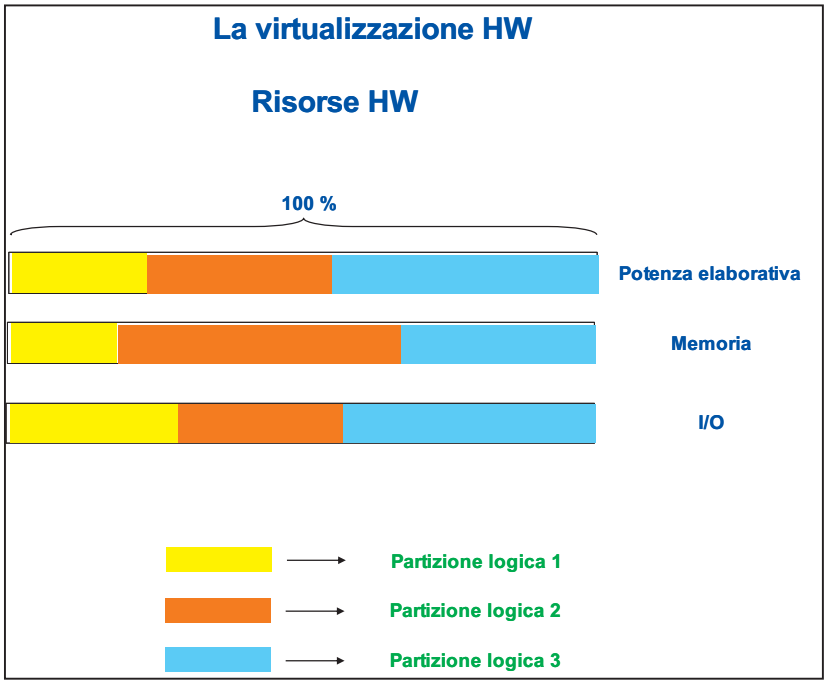


Figura 4 Esempio di virtualizzazione

Ogni partizione è governata da una singola istanza di sistema operativo. E' importante notare che nelle tre partizioni descritte in Figura 4 i sistemi operativi possono essere di tipo diverso. Inoltre una partizione può essere attivata o disattivata senza influenzare le partizioni adiacenti. Avendo introdotto il concetto di partizione logica, possiamo stabilire che tutte le volte che in questo libro parleremo di server o sistema, in realtà parleremo di una partizione logica secondo la definizione di cui sopra, a meno di una esplicita definizione contraria.

I dispositivi di I/O sono connessi al channel subsystem attraverso le unità di controllo (control unit). La connessione tra il channel subsystem e una unità di controllo è chiamata *channel path;* esso può utilizzare o un protocollo di trasmissione parallela (non più supportato nei dispositivi attuali) o un protocollo di trasmissione seriale, abbiamo perciò channel path paralleli o seriali. Un Channel Path parallelo può connettere più Control Unit. Un Channel Path seriale può connettere una sola Control Unit (Topologia Point to Point). La connessione a più Control Unit si realizza utilizzando dei dispositivi chiamati switch dinamici.

Oltre alla memoria principale è prevista dall'architettura la memoria espansa, ovvero una memoria non indirizzabile direttamente da un programma applicativo, ma utilizzabile dal sistema operativo a blocchi di 4KB come memoria veloce di paging. La

memoria espansa è definita dalla configurazione come parte della memoria fisica di una macchina.[2] Nella CPU può essere inclusa un'unità crittografica e/o una fonte esterna di sincronizzazione (External Time Reference - ETR).

La natura fisica delle funzioni citate sopra può variare tra le implementazioni hardware, chiamate "modelli". La Figura 3 mostra la struttura logica di un sistema multiprocessore a 2 CPU con un'unità crittografica e che è connesso a un ETR. Ogni modello di mainframe ha una sua configurazione, definita dall'insieme delle funzioni disponibili e delle risorse HW, quali il numero di canali, dei channel path, delle unità di controllo che si possono connettere al channel subsystem, la dimensione della memoria principale ed espansa e le modalità di gestione della configurazione.

La gestione della memoria

Poiché la z/Architecture si rifà al modello di von Neumann, è importante comprendere bene come viene vista la memoria dall'architettura. Il concetto fondamentale è che la memoria (secondo il modello di von Neumann) nell'architettura z è **memoria virtuale:** è cioè un'astrazione ad uso di chi scrive SW (di qualunque livello, sistema operativo, database manager e così via). La dimensione di questa memoria è limitata solo dalla capacità dell'indirizzamento. Il concetto di memoria virtuale è presente in molte architetture. I dettagli implementativi e il supporto hardware differenziano le architetture. Utilizzando la memoria virtuale ogni programma in esecuzione può assumere di avere accesso a tutta la memoria definita dallo schema d'indirizzamento dell'architettura. Il solo limite è il numero di bit che compongono l'indirizzo secondo l'architettura. Oggi la z/Architecture prevede tre diverse modalità di indirizzo (indirizzamento trimodale): 24, 31 e 64 bit. Quindi avremo:

- 24 bit = 16 MB
- 31 bit = 2 GB
- 64 bit = 16 Exabyte

In ogni istante ogni CP può trovarsi in una delle tre modalità e, nell'ambito dello stesso sistema, i CP possono avere nello stesso istante modalità di indirizzo differenti. L'addressing mode è governato da due bit nella Program-Status Word (PSW – vedere più avanti). Poter usare per programmi e dati più memoria di quella realmente disponibile presenta una serie di vantaggi. Il primo e più evidente è che si possono mettere sulla stessa macchina applicazioni che richiedono molta più memoria di quella realmente disponibile. L'implementazione della memoria virtuale permette di ottenere questo apparente miracolo con una perdita minima di prestazioni.

I sistemi operativi come z/OS, zLinux e z/VM supportano indirizzi virtuali e reali a 64 bit, che consentono a un programma di indirizzare fino a 16 Exabyte di locazioni di memoria. Nella realtà i server oggi disponibili a 64 bit (di qualunque architettura) non

[2] La memoria espansa fu originariamente introdotta per superare alcuni limiti dovuti alla modalità di indirizzamento a 31 bit. Essendo la z/Architecture a 64 bit essa non è più necessaria e non è attualmente utilizzata dal sistema operativo z/OS. z/VM la utilizza come memoria di paginazione di primo livello per ottimizzare la gestione della memoria virtuale.

si spingono oltre pochi Terabyte di memoria reale. I sistemi operativi tengono in memoria centrale le porzioni attive del programma e dei dati e, secondo un algoritmo specifico, trasferiscono quelle inattive tra la memoria centrale e quella ausiliaria su disco, se richiesto dall'integrità dell'elaborazione.

La memoria virtuale che il sistema operativo assegna a ogni processo per suo uso esclusivo viene chiamata Address Space (AS); esso è un insieme di indirizzi virtuali contigui disponibile per eseguire istruzioni e immagazzinare i dati. Questo insieme può essere tanto grande da comprendere tutto l'intervallo indirizzabile secondo l'architettura. Quindi un Address Space in architettura a 64 bit può comprendere fino a 16 Exabyte. Questa è la dimensione di ogni Address Space nei mondi z/OS, z/VM e zLinux.

Poichè l'architettura prevede che tutti gli indirizzi generati dai processori siano virtuali, essi devono essere sottoposti al processo di traduzione Virtuale → Reale. Viceversa, gli indirizzi di memoria generati dai canali e utilizzati durante l'esecuzione delle operazioni di I/O (ad esempio quelli che individuano le aree dove scrivere i dati provenienti dagli I/O device) sono reali e non sono tradotti dall'Hardware. Questa diversità di trattamento è conseguenza del fatto che i dispositivi di I/O sono più sensibili alla tempificazione delle operazioni e quindi non possono tollerare i ritardi che la traduzione dinamica potrebbe generare.

La traduzione dinamica degli indirizzi (DAT) è il processo di traduzione di un indirizzo virtuale nel corrispondente indirizzo reale eseguito durante l'accesso alla memoria. Questa traduzione usa delle tabelle (genericamente indicate come Page Table) che indicano per ogni pagina se questa risiede in una frame o no. Se la traduzione ha successo, l'esecuzione dell'istruzione che ha richiesto l'accesso prosegue. In caso contrario, si verifica un'interruzione per page fault, il sistema operativo viene avvisato, viene lanciata l'operazione di caricamento della pagina dalla memoria ausiliaria e il processo SW interessato viene sospeso in attesa che il caricamento della pagina si compia. Le Page Table sono lette dal CP durante il processo di traduzione e sono costruite e gestite dal SW. Poiché il processo di traduzione è abbastanza oneroso, per velocizzarlo si usano delle memorie di tipo associativo chiamate Translation Lookaside Buffer (TLB) in cui vengono memorizzate le ultime "n" traduzioni Virtuale → Reale andate a buon fine; la grandezza di "n" è uno dei fattori più importanti per le prestazioni del processore.

Per il programmatore, l'intero programma sembra occupare uno spazio contiguo di memoria. Di fatto non tutte le pagine di un programma sono necessariamente nella memoria centrale o occupano spazio contiguo. Dalla Figura 5 si vede che per designare una posizione di memoria virtuale il solo indirizzo non e' sufficiente: bisogna anche indicare quale AS contiene quell'indirizzo. Nell'architettura questa identificazione prende il nome di Address Space Number (ASN). Sono possibili 64K ASN per sistema (cioè per Partizione Logica).

Le parti di un programma che vengono eseguite in memoria virtuale devono essere mosse tra la memoria reale e quella ausiliaria. Per permettere questo i sistemi operativi gestiscono la memoria in unità, o blocchi, di 4KB.

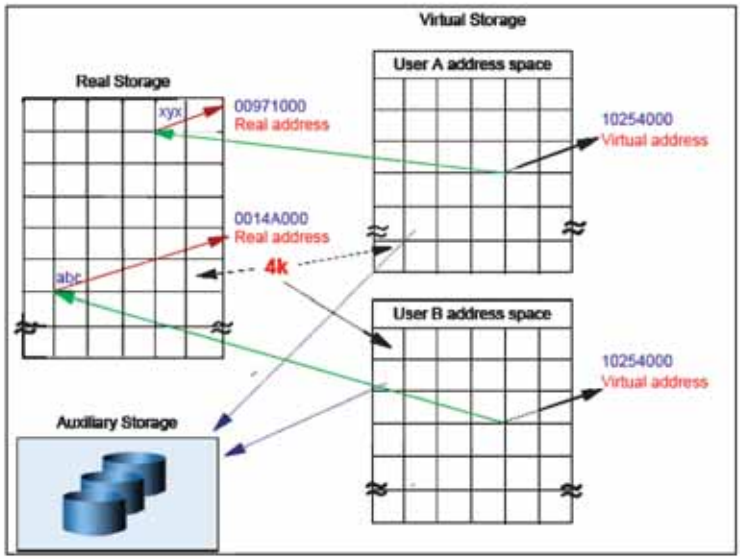


Figura 5 Real e Auxiliary storage per la memoria virtuale

Sono definiti i seguenti blocchi:

- Un blocco di memoria centrale è una *frame*.
- Un blocco di memoria virtuale è una *pagina.*
- Un blocco di memoria ausiliaria è uno *slot*.

Frame, pagine e slot hanno la stessa dimensione: 4KB (vedi figura 6). Per la relazione fra Page, Frame e Slot si rimanda alla descrizione del sistema operativo z/OS.

La traduzione dell'indirizzo da Virtuale a Reale avviene in modo molto semplice secondo la Figura 7. L'esempio fa riferimento a un Address Space da 2 GB (cioè siamo in modalità 31 bit). In un AS da 16 Exabyte il numero di tabelle da percorrere per tradurre l'indirizzo aumenta (si hanno altre tre tabelle di livello superiore (Region I table, Region II table, Region III table).

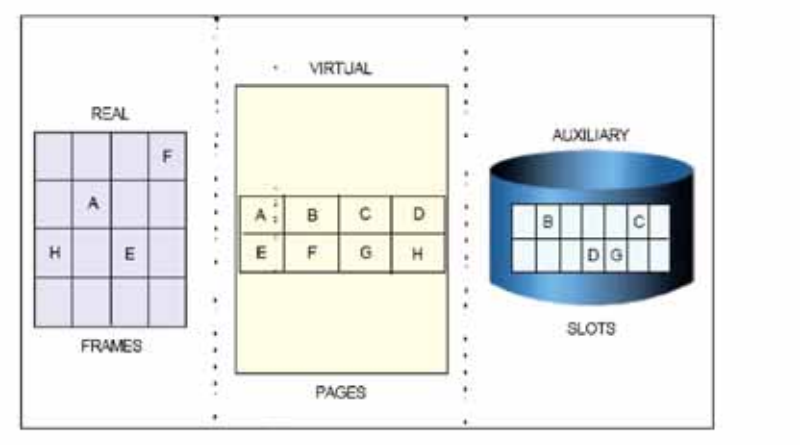


Figura 6 Frame, Page e Slot

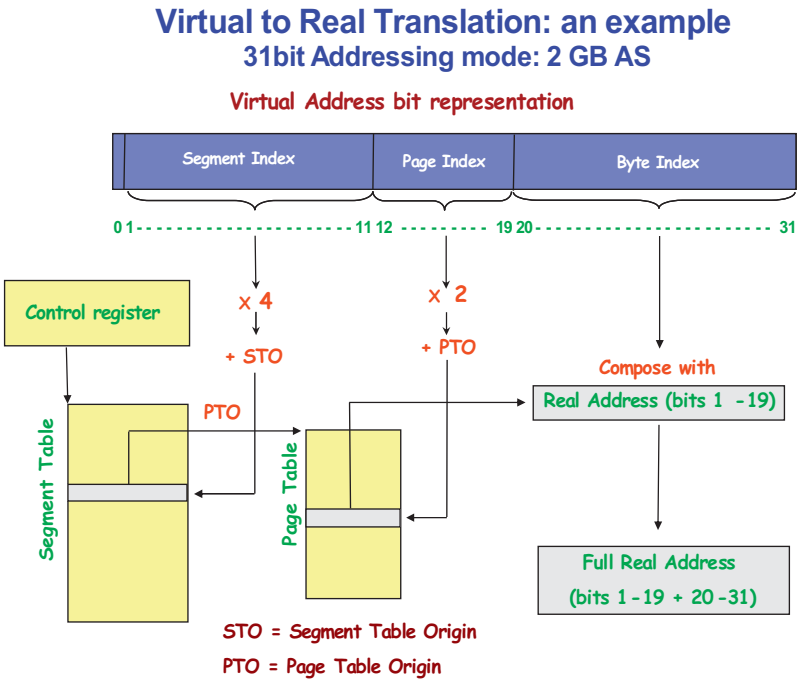


Figura 7 Il processo di traduzione dinamica degli indirizzi

Segment Table Origin (STO) indica l'indirizzo in Memoria Reale (domanda per il lettore: perchè non è Memoria Virtuale?) della Segment Table da cui far partire la traduzione.

STO è contenuto in uno specifico Control register (per la definizione di Control register vedere più avanti), definito dall'architettura. Ogni AS attivo nel sistema (cioè nella Partizione Logica) ha una sua Segment table a cui è associato un Segment Table Origin. Caricando quindi un STO in quel Control Register, la traduzione si muoverà nell'AS rappresentato da quella Segment Table e di conseguenza anche l'esecuzione del SW avverrà all'interno di questo AS.

In questo algoritmo c'è bisogno di un accesso in memoria per ogni livello di tabella da consultare. Quindi in un AS da 64 bit con cinque livelli di tabelle da percorrere si deve accedere cinque volte alla memoria. E' facile capire che questo rappresenta un peso inaccettabile per l'elaborazione. Da qui la necessità di un buffer che memorizzi le traduzioni più recenti, in modo da evitare il più possibile di ricorrere all'algoritmo standard. Questo buffer esiste ed è il TLB. La sua presenza è teoricamente opzionale ma assolutamente necessaria se si vuole che il mainframe abbia le prestazioni desiderate.

Per supportare la traduzione Virtuale → Reale in un AS da 64 bit la quantità di memoria (reale) per ospitare i cinque livelli di tabelle sarebbe, senza particolari accorgimenti, pari a parecchie centinaia di Megabyte. L'architettura mette a disposizione dei sistemi operativi una serie di controlli (posizionati nei Control register di cui parleremo più avanti) in modo da ridurre queste quantità al minimo indispensabile. Ad esempio, per un AS da 2 GB (e praticamente tutti gli AS oggi sono da 2 GB, salvo quei rari casi in cui qualche applicazione vuole usare memoria sopra i 4 GB), basteranno pochi KB (reali) per tradurre gli indirizzi.

Il controllo degli accessi

L'architettura mette a disposizione del SW dei meccanismi di protezione per garantire l'integrità delle elaborazioni. Ad ogni pagina sono associati 7 bit che costituiscono la Storage Key:

- Storage protection key (4 bit; valori possibili: da x'0' a x'F')
- Fetch Protection bit (si vuole controllare l'accesso in lettura)
- Reference bit (almeno un byte nella pagina è stato referenziato)
- Change bit (almeno un byte della pagina è stato modificato).

La Storage protection key viene usata ogni volta che l'esecuzione del SW genera un accesso alla pagina in scrittura (ad esempio il risultato di una somma deve essere messo in un operando nella pagina). Esistono nella PSW (vedere più avanti) 4 bit che costituiscono la Access Key: se il valore della Storage protection key è uguale all'Access key nella PSW la scrittura è permessa; in caso contrario il programma viene interrotto con un'interruzione di Program Check. Se il bit di Fetch Protection è ON allora lo stesso controllo viene fatto anche per gli accessi in lettura. Un caso particolare si ha quando il valore dell'Access key è x'0': in questo caso il controllo non viene eseguito e l'accesso viene effettuato comunque. Si dice che il programma sta eseguendo in chiave zero.

Esistono poi due meccanismi di protezione che si appoggiano alla traduzione dinamica degli indirizzi:

- è possibile indicare se una certa pagina può essere acceduta mettendo in ON un certo bit nella Page Table Entry corrispondente;
- è possibile restringere l'accesso ad una pagina alla sola lettura mettendo in ON un altro bit nella Page Table Entry corrispondente.

In entrambi i meccanismi il programma terminerà con Program Check. Reference e Change bit sono usati dai sistemi operativi per ottimizzare l'uso della memoria reale. La Storage Key è un'entità HW associata ad ogni Frame; ma il valore da dare alla Storage protection key (i primi 4 bit) viene impostato in base al valore che il sistema operativo dà alla pagina virtuale che occupa quella Frame.

I registri e le istruzioni

La z/Architecture è classificata tra le architetture CISC, poiché rende disponibili molte istruzioni complesse di lunghezza variabile. Il numero di registri è inferiore a quello dei registri delle architetture RISC. Ogni CP possiede un set completo di registri. Essi si dividono in:

- **Registri Generali (General Register)**: sono 16 registri a 64 bit; per avere compatibilità all'indietro con le applicazioni scritte per le architetture precedenti, tali registri possono essere usati in parte come registri a 32 bit; possono essere utilizzati per indirizzare posizioni di memoria e come accumulatori nella aritmetica generale e nelle operazioni logiche.
- **Registri per la Virgola Mobile (Floating Point Register)**: sono 16 registri a 64 bit; sono utilizzati per le operazioni in virgola mobile (binarie ed esadecimali); possono essere adoperati anche a coppie per contenere operandi a 128 bit.
- **Registri di Controllo (Control Register):** sono 16 registri da 64 bit usati dai CP per controllare l'esecuzione delle istruzioni. Ogni bit in questi 16 registri ha una funzione assegnata dall'architettura. Manipolando questi registri si governa l'esecuzione del SW in un CP.
- **Registri d'accesso (Access Register):** permettono l'accesso ai Data Space (Address Space che contengono solo dati).

I formati delle istruzioni

Il **set di istruzioni CISC** nella z/Architecture è molto ricco: ci sono 21 Formati per parecchie centinaia di istruzioni diverse. Ogni istruzione è costituita da due parti:

- **Operation Code:** specifica quale operazione deve essere eseguita.
- **Operand Designation:** può essere l'indirizzo del dato che si deve elaborare, la sua lunghezza o il dato stesso.

Un'istruzione può essere lunga una, due o tre halfword (ovvero 16, 32 o 48 bit) come mostrato nella figura seguente e deve essere contenuta in memoria allineata alla halfword.

Ogni istruzione è in uno dei 18 formati base E, RR, RRE, RRF, RX, RXE, RXF, RS, RSE, RSL, RSI, RI, RIE, RIL, SI, S, SSE, e SS, con 3 variazioni del formato RRF, due dei formati RS, RSE e RIL, e 4 del formato SS; la figura 8 illustra alcuni di questi formati.

L'Instruction Set

L'insieme delle istruzioni della z/Architecture (al 2008 sono 894 di cui 668 implementate direttamente in hardware) ne costituisce l'Instruction Set. Esso è un elemento chiave di ogni architettura. Coloro che scrivono Software per una certa unità centrale dipendono dall'Instruction Set direttamente (mediante il linguaggio Assembler) o indirettamente (mediante linguaggi di alto livello come Cobol o Java): il processo di compilazione trasforma la descrizione della logica di business descritta nel linguaggio scelto in una sequenza di istruzioni Hardware appartenenti tutte all'Instruction Set; questa sequenza viene memorizzata come file binario su disco in apposite librerie.

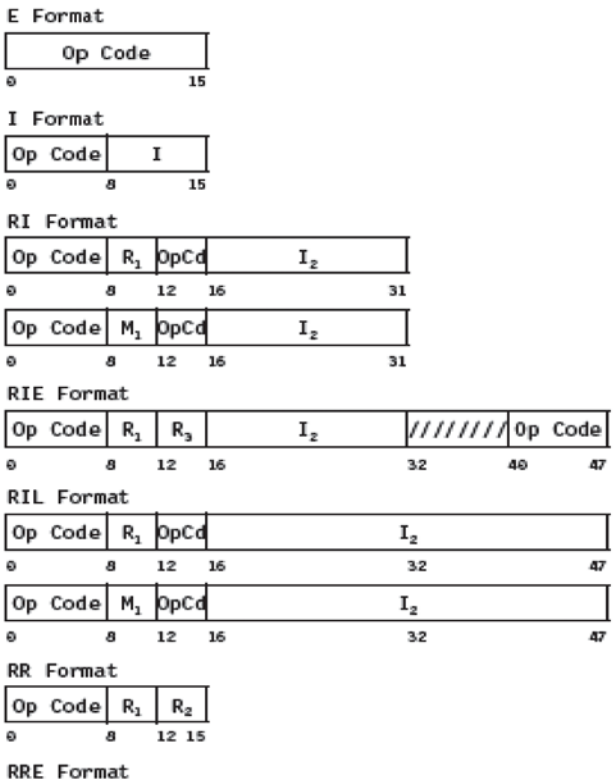


Figura 8 Basic Instruction Format

Questo file binario viene comunemente chiamato eseguibile o Load Module (in z/OS). Perciò ogni linguaggio ha tanti compilatori quante sono le piattaforme su cui può essere usato: così abbiamo un compilatore "C" per la architettura z, uno per Linux, uno per AIX ecc.

Se l'Instruction Set evolve solo aggiungendo istruzioni e mai levandole si ottiene una delle caratteristiche fondamentali della z/Architecture: la compatibilità binaria all'indietro. Infatti l'Instruction Set aggiunge sempre istruzioni, non ne elimina mai; quindi programmi compilati alla nascita del mainframe (1964) possono essere usati ancora oggi senza bisogno di essere ricompilati; questo è uno dei requisiti fondamentali per garantire la compatibilità binaria del codice eseguibile (upward compatibility). L'Instruction Set è un elemento chiave dell'architettura ed è formato da tutte le istruzioni che la CPU è in grado di eseguire.

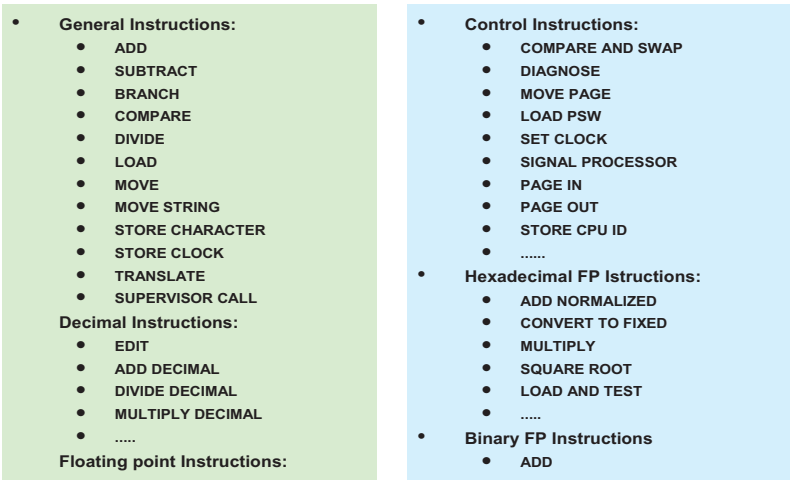


Figura 9 Principali istruzioni della z/Architecture

Abbiamo visto che per mantenere la compatibilità binaria con i livelli delle architetture precedenti, è previsto l'indirizzamento trimodale; esso si riferisce alla capacità di commutare tra i modi di indirizzamento a 24, 31 o 64 bit. La commutazione può essere fatta o con le vecchie istruzioni di BRANCH che impostano il modo sulla base di un registro di controllo, o mediante la nuova istruzione SET ADDRESSING MODE (SAM24, SAM31, e SAM64).

Lo stato di un CP può essere PROBLEM o SUPERVISOR. Non tutte le istruzioni possono essere eseguite in Problem state, ma tutte possono essere eseguite in Supervisor state. Ad esempio Load Control (l'istruzione che cambia il contenuto di un Control Register) è un'istruzione privilegiata, esige cioè il Supervisor state. Si passa da Problem state a Supervisor state eseguendo l'istruzione Supervisor Call (SVC) il cui unico argomento è un numero da 0 a 255. L'esecuzione di questa istruzione genera un

Interrupt (sincrono poiché è previsto dal programma) che mette il CP in Supervisor state. Il numero specificato diventa l'Interruption code.

Un esempio chiarisce l'uso di questa istruzione: in z/OS se si vuole aprire un Data Set si esegue "SVC 19". SVC First Level Interruption Handler (SVC FLIH – la routine che prende il controllo subito dopo l'interruzione) vede il valore 19 e passa quindi il controllo al componente del sistema operativo che apre il Data Set. Quando il Data Set è aperto l'esecuzione riprende dalla istruzione immediatamente successiva alla "SVC 19".

La Program-Status Word (PSW)

La PSW è un circuito di memoria collocato all'interno di ogni CP (Central Processor). Essa contiene le informazioni richieste per l'esecuzione del programma attivo nel CP: contiene lo stato istantaneo (cioè ciclo per ciclo) di un CP; ad esempio in una configurazione con 13 CP avremo 13 PSW, una per CP. Abbiamo già visto alcune informazioni contenute nella PSW: Addressing mode, Access key, e stato del CP. Un'altra informazione importante contenuta nella PSW è se il CP lavora o è in wait. Quando un CP è in wait cessa di eseguire istruzioni. La lunghezza è di 16 byte (128 bit). La PSW include l'indirizzo della prossima istruzione da eseguire, un codice in cui viene indicato il risultato dell'istruzione che viene eseguita (Condition Code) e altre informazioni usate per controllare la sequenza di esecuzione delle istruzioni e per determinare lo stato della CPU. La PSW viene anche indicata come Current PSW per distinguerla da due copie in memoria virtuale che vengono usate nella gestione degli Interrupt: Old PSW e New PSW. La relazione fra Current PSW, Old PSW e New PSW viene illustrata in seguito.

In sintesi possiamo dire che Control register e PSW guidano l'esecuzione del SW sui CP. La differenza fondamentale fra le due entità è che il contenuto dei Control register cambia a fronte dell'esecuzione di specifiche istruzioni (e quindi è governato dal SW), mentre il contenuto della PSW cambia ogni volta che viene eseguita una istruzione, a fronte dell'esecuzione di specifiche istruzioni o a fronte di specifici eventi HW (come le interruzioni).

Le Interruzioni

L'Interruzione o Interrupt è un evento, codificato dall'architettura, che può essere indipendente dall'esecuzione del Software e quindi asincrono, o legato all'esecuzione del Software e perciò sincrono; l'effetto di questi eventi è di alterare la sequenza di istruzioni a seconda del tipo di evento (Interrupt) che si è verificato.

Quando avviene un'Interrupt, la CPU salva la Current PSW in una locazione di memoria assegnata (denominata appunto Old PSW); ogni tipo di Interrupt ha una sua locazione Old PSW la cui posizione in memoria reale è codificata dall'architettura. Contestualmente 16 byte vengono caricati nella current PSW dalla locazione indicata come New PSW. Come per le Old PSW così esiste una New PSW per ogni tipo di Interrupt. Poiché una PSW contiene l'indirizzo della prossima istruzione da eseguire, la New PSW per un certo tipo di Interrupt conterrà l'indirizzo della prima istruzione eseguibile della routine Software che dovrà gestire l'Interrupt. La Old PSW salvata

servirà a riprendere l'esecuzione del processo Software che era stato interrotto partendo dall'indirizzo presente nella Old PSW che appunto indica la prima istruzione da eseguire dopo l'Interrupt.

Ci sono sei classi di interruzione; ogni classe ha una coppia di locazioni per Old e New PSW la cui posizione in memoria reale è, come si è visto in precedenza, stabilita dall'architettura. Le sei classi di interruzione sono:

- External: eventi scollegati dall'esecuzione del Software e non legati alle operazioni di I/O;
- Machine Check: è avvenuto un malfunzionamento a livello di Hardware;
- I/O: una operazione di I/O si è conclusa;
- Restart: un operatore richiede l'esecuzione di una certa routine Software il cui indirizzo è contenuto nella New PSW;
- Supervisor Call: il programma chiede l'esecuzione di una particolare routine fornita dal Sistema Operativo, ad esempio il lancio di un'operazione di I/O – l'implementazione di questo tipo di Interrupt è dipendente dal sistema operativo;
- Program Interruption: l'esecuzione del programma non può proseguire perché si sono verificate delle condizioni la cui soluzione richiede l'intervento del sistema operativo; ad esempio è necessario portare in memoria reale dai data set di page una pagina di memoria virtuale, o si è tentata una divisione per zero.

1.1.2 Architettura di I/O

La connessione Applicazione - Dati

In generale ogni applicazione ha bisogno di scambiare dati con fonti esterne, che possono essere basi dati o il Network. Questo viene reso possibile collegando il server con i dispositivi storage (dischi, nastri) o con i dispositivi di rete. Nell'architettura mainframe questa connessione è particolarmente ricca di funzioni e si articola in una definizione architetturale e una implementazione reale.

Discuteremo l'implementazione reale nella descrizione del modello z10 in seguito.

Definizione architetturale

I due capi della connessione sono da un lato il server e dall'altro il dispositivo di I/O (I/O device nella definizione architetturale). Fra il server e l'I/O device esistono due entità molto ben definite architetturalmente nelle rispettive funzioni e ruoli che però nell'implementazione possono addirittura ridursi a delle routine SW:

- Il Canale (Channel)
- L'Unità di Controllo (Control Unit)

Nella figura 10 vediamo un esempio di questa connessione applicata al caso di un sistema storage.

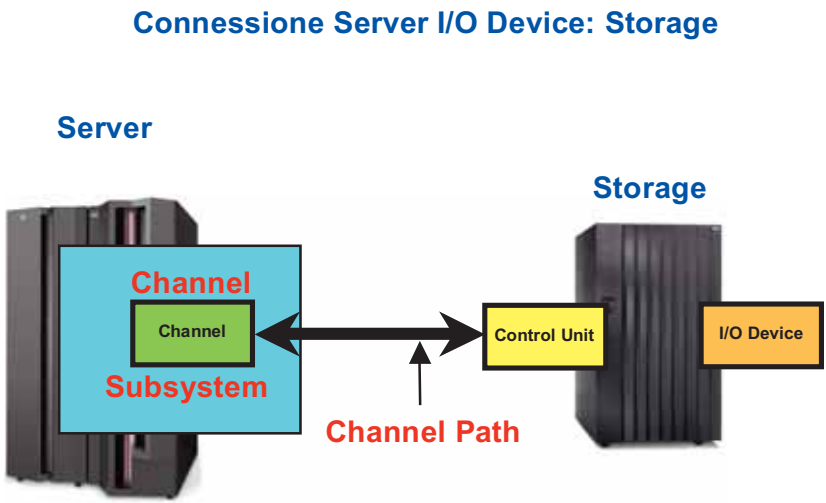


Figura 10 Connessione Server e I/O device

In questo caso il server e l'I/O Device Storage stanno in box separate; l'entità architetturale Control Unit è implementata all'interno della box dell'I/O device, mentre l'entità architetturale Canale è implementata all'interno della box Server. Il canale è sempre realizzato all'interno del server. Control Unit e I/O Device (si noti che una Control Unit può controllare più di un I/O Device come nel caso dei sistemi storage) sono quasi sempre implementati insieme in complessi HW separati dal server; fanno eccezione gli adapter di rete che, trovandosi all'interno del server, hanno le tre funzioni (Canale, Control Unit e I/O Device) realizzate all'interno del server.

E' una struttura piramidale: un server può avere fino a 256 canali, ogni canale fino a 256 control unit, ogni control unit fino a 256 I/O Device. Ogni server può accedere a non più di 65536 I/O Device.

I sistemi operativi che operano sul mainframe devono essere in grado di gestire canali ed I/O device; per poter fare questo si basano su due identificazioni definite dall'architettura:

- Canale: è identificato da "Channel Path Identifier – CHPID" (il limite di 256 canali nasce dal fatto che il CHPID occupa un Byte);
- I/O Device: è identificato da due entità:
    - Device Number (il limite di 65536 I/O device per server deriva dal fatto che il Device Number occupa due byte); il Device Number è

associato all'I/O device con un processo di definizione chiamato I/O Configuration Program (IOCP);
- Subchannel Number è l'identificazione del blocco di controllo che all'interno del Channel subsystem descrive un certo I/O Device. La sua funzione è duplice: raccoglie le informazioni che l'IOCP ha costruito sull'I/O Device (fra queste ci sarà ad esempio il Device Number relativo) e registra lo stato del device durante l'esecuzione delle operazioni di I/O.

L'architettura chiama la parte di un server mainframe che sovrintende al traffico di I/O con il nome di Channel Subsystem. Le sue funzioni principali sono:

- Controllare tutti i canali presenti nella configurazione. Il controllo è di due tipi: durante l'esecuzione delle operazioni di I/O che il SW richiede e per reagire ad eventi estemporanei, quali un guasto di una fibra ottica o un comando operativo che vuole modificare qualcosa nella configurazione.
- Ricevere dal SW (tipicamente il componente del sistema operativo che interfaccia il Channel subsystem) la richiesta di eseguire una operazione di I/O. In termini architetturali il SW esegue l'istruzione Start Subchannel indicando la posizione in memoria (Reale) del programma di canale che deve essere eseguito.
- Presentare ad un CP l'I/O interrupt che segnala la fine dell'esecuzione del programma di canale e quindi dell'operazione di I/O. Mediante il meccanismo dello scambio delle PSW il sistema operativo verrà coinvolto in modo da gestire l'evento e notificare all'applicazione che aveva richiesto l'operazione di I/O il suo completamento.
- Poiché fino a 8 Channel Path possono essere usati su un singolo I/O device, è compito del Channel subsystem scegliere un Channel Path su cui eseguire l'operazione di I/O verso il device.
- Il Channel subsystem decide come gestire il caso in cui due o più richieste di I/O siano attive sullo stesso I/O device; può avvalersi di suoi algoritmi interni o può essere istruito dal sistema operativo su quale operazione sia più importante (ad esempio considerando l'importanza che la transazione che ha richiesto l'operazione ha per il Business aziendale).

Poiché Server ed I/O Device possono essere fisicamente separati, è necessario collegarli affinchè lo scambio di dati possa avvenire: questa connessione è realizzata da un cablaggio (elettrico o ottico) che prende il nome di Channel Path. Oggi la connessione usata è ottica e lo standard su cui è basata viene chiamato Fiber Channel (FC Standard Architecture). Il protocollo di comunicazione fra Channel e Control Unit fa parte dell'architettura del mainframe (è descritta nel "Principles of Operations"), mentre lo scambio di segnali lungo la fibra ottica fa parte dello standard FC. La tecnologia consente oggi di avere fra canale e Control Unit (o fra server ed I/O device) distanze di decine di chilometri.

Alcuni principi fondamentali sul funzionamento dell'I/O device:

- Ogni I/O device può essere collegato ad un server con più di un channel path: l'attuale limite architetturale è 8 channel path. Questa funzione è molto utile perché garantisce l'accesso all'I/O device anche in caso di perdita di uno o più channel path; inoltre, se il numero di byte da trasferire

è tale da superare la capacità di un Channel path, potendone configurare fino ad 8 si garantisce un'ottima scalabilità dal punto di vista del Data Transfer Rate.

- Un I/O device può essere collegato a più di un server (vedere figura 11). Il numero di server che può essere contemporaneamente collegato è variabile e dipende dalla implementazione del device.
- Un I/O device può eseguire una sola operazione di I/O. E' quindi possibile che venga chiesto di fare una operazione di I/O ad un device che è impegnato con un'altra operazione di I/O: per gestire questo evento esiste il concetto di Device Busy, uno stato che deve essere riferito al server per gestire la situazione (cioè accodare la richiesta per rilanciarla in tempi migliori). Per gestire questa situazione requisito essenziale è che il Device segnali che l'operazione in cui era impegnato è terminata e che adesso può accettare nuove richieste.

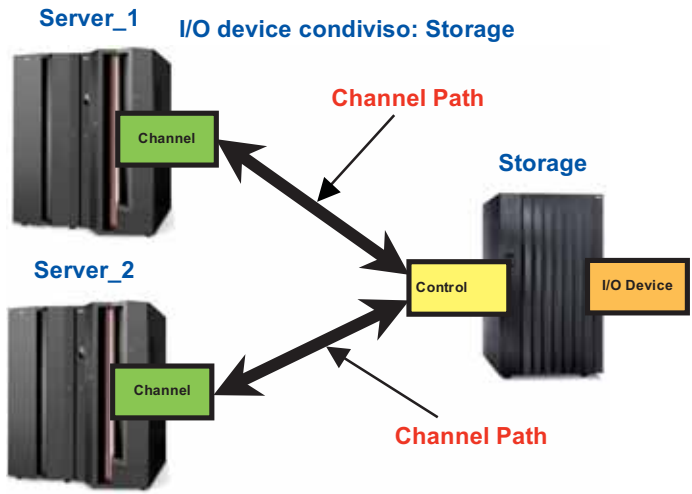


Figura 11 Un I/O device condiviso fra due server

Per poter realizzare configurazioni di I/O così ricche di possibilità c'è bisogno di unità di switch che permettano di moltiplicare le possibilità di collegamento fra Canale e Control Unit. Dobbiamo tener presente infatti che la connessione in Fibra Ottica è per sua natura Point to Point (il collegamento Arbitrated Loop non è praticabile per configurazioni così complesse come si trovano nelle installazioni Mainframe). Nella definizione dello standard Fiber Channel una parte essenziale è la descrizione delle funzioni e del protocollo di questo switch. Nella figura 12 vediamo una configurazione con due server e tre sistemi storage connessi mediante uno Switch Fiber Channel.

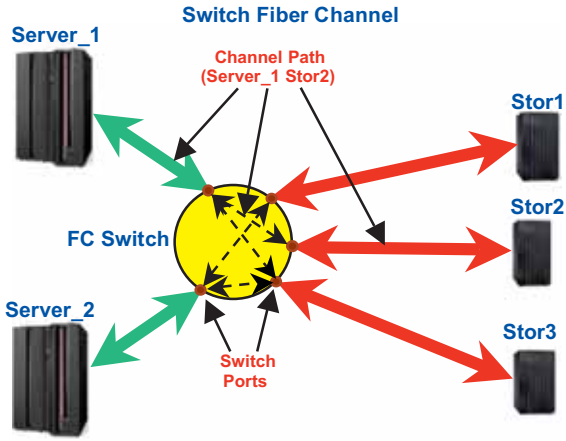


Figura 12 Configurazione di I/O con uno switch FC

In questa figura vediamo che:

- **Server_1** ha stabilito delle connessioni con **Stor2** e **Stor3**
- **Server_2** ha stabilito delle connessioni con **Stor1** e **Stor3**
- La nozione Channel Path viene estesa e, in presenza di uno switch, il Channel Path comprende:
  - La connessione Canale – Switch (segmento verde)
  - La connessione fra le due porte dello switch (segmento tratteggiato nero)
  - La connessione fra lo switch e la Control Unit (segmento rosso)
- Le connessioni all'interno dello switch non sono fisiche ma logiche, e fanno parte della definizione della configurazione di I/O

La comunicazione fra Control Unit e I/O Device non fa parte dell'architettura mainframe e quindi, ad esempio, ogni Sistema Storage collegabile al mainframe realizza questa connessione in modo da ottimizzare il proprio funzionamento interno e lo scambio di dati fra l'I/O Device ed il Mainframe. La Control Unit può governare più di un I/O device, deve quindi avere il modo di individuare il Device che deve eseguire l'operazione che il canale richiede. Perciò esiste un indirizzamento interno alla Control Unit che prende il nome di Unit Address. Questo Unit Address deve essere accoppiato al Device Number associato all'I/O device. In seguito verrà illustrato il modo in cui questa e altre definizioni vengono stabilite.

Ogni I/O Device effettua lo scambio di dati con il mainframe eseguendo dei comandi; tali comandi gli sono impartiti dal canale. Un'operazione di I/O è realizzata mettendo in sequenza questi comandi. Questi comandi sono simili alle istruzioni HW che il SW esegue; le differenze sono:

- Sono eseguite dagli I/O device e non dai processori;
- L'architettura impone il formato del comando: ad esempio un comando di lettura (cioè un comando la cui esecuzione implica che i byte passino dal device al server) avrà un certo bit in OFF, mentre un comando di scrittura (cioè un comando la cui esecuzione implica che i byte passino dal server al device) avrà lo stesso bit in ON. L'esatta natura del comando dipenderà dalle particolari funzioni che il device è in grado di svolgere. Ad esempio, un disco avrà un comando per spostare il braccio che ospita le testine di lettura/scrittura in un punto preciso della superficie (comando di Seek), mentre un'unità a nastro avrà un comando per fare il rewind della cartuccia.

Supponiamo di dover leggere un record da un database. La sequenza di comandi sarà costituita da tre comandi (useremo i termini usati dal sottosistema storage IBM DS8000):

1. **Define Extent** - Impostare i parametri generali dell'operazione (Lettura; posizione nell'ambito dell'archivio del record da leggere; opzioni di caching all'interno del sistema Storage, e altro)
2. **Locate record** – Posizionare logicamente l'I/O device sul record da leggere
3. **Read record** - Leggere il record e trasferirne il contenuto al canale.

Quando l'esecuzione termina, un particolare evento (Channel End) genera un I/O interrupt dal canale verso la CPU (e quindi verso il SW). Questa sequenza di comandi prende il nome di **Programma Canale.** Il programma canale è preparato dal SW: ad esempio l'accesso ai Database è preparato dal Media Manager dello z/OS; scambiare informazioni con gli utenti attraverso il Network è gestito dal Device Driver degli adattatori di rete (un componente del TCP/IP).

Prendendo ad esempio il programma di canale sopra descritto vediamo come lo si esegue (vedere figura 13):

1. Il primo comando (Define Extent) è passato dal canale alla Control Unit;
2. La Control Unit riceve dal canale, insieme al comando, le informazioni correlate ed esegue il comando;
3. La Control Unit segnala al canale che l'esecuzione è terminata e se l'esito è positivo o negativo; genera cioè un evento che viene notificato al canale insieme ad una serie codificata di informazioni (Status bytes) su come l'esecuzione di questo comando è terminata (bene, male, con quali errori). Si noti che l'esecuzione del comando può coinvolgere o no l'I/O device (in questo caso il disco su cui si trova il record del Database che deve essere letto). Il comando Define Extent non coinvolge l'I/O device;
4. Il Canale verifica le informazioni passate dalla Control Unit e, se lo status indica che l'esecuzione del comando è andata a buon fine, passa al comando successivo (Locate record);
5. La Control Unit passa questo comando all'I/O Device per l'esecuzione; viene eseguito dal Device che indica alla Control Unit l'avvenuta esecuzione;
6. La Control Unit segnala l'esecuzione corretta del comando di Locate record;
7. Il Canale, ricevuta dalla Control Unit la notifica che il comando è stato eseguito e lo status è positivo, passa alla Control Unit il comando successivo (Read - è anche l'ultimo);

8. La Control Unit esegue questo comando; anche in questo caso l'I/O device è coinvolto poiché bisogna trasferire il contenuto del record dal Database al server;
9. La fine dell'esecuzione di questo comando indica anche che l'operazione di I/O si è completata (il programma di canale è stato eseguito);
10. Il Canale, sapendo che il comando eseguito era l'ultimo, genera un I/O interrupt in modo da informare il sistema operativo che l'operazione di I/O è terminata.

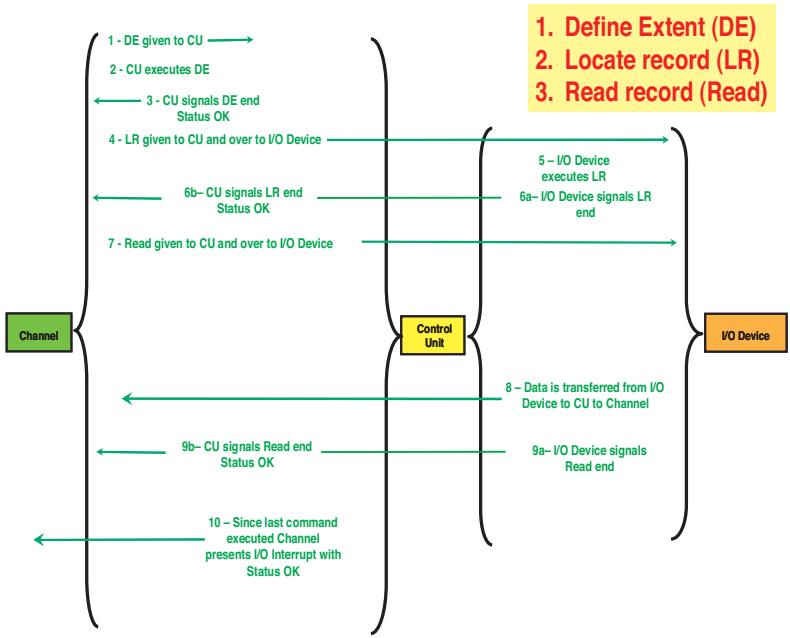


Figura 13 Esecuzione di un programma di canale (Storage)

Da questa sequenza impariamo varie cose. Il sistema operativo lancia l'operazione e riceve la notifica che l'operazione è terminata (I/O interrupt). Se però l'esecuzione di un comando intermedio non va a buon fine, l'esecuzione si interrompe e l'I/O interrupt che viene generato indica al SW (mediante gli status byte) quale comando non ha funzionato e perché; a questo punto il SW è in grado di invocare le routine di recovery appropriate.
Il programma che abbiamo descritto è costruito per leggere un record, ma dei tre comandi che lo compongono solo il terzo è un comando di lettura, i primi due sono comandi di scrittura; il motivo è semplice: Lettura/Scrittura è un concetto che, a livello del singolo comando, si applica al senso in cui sono trasferiti i dati. Poiché i primi due

comandi devono istruire la Control Unit sulla natura dell'operazione, di fatto trasferiscono byte verso la Control Unit e sono perciò comandi che scrivono.

L'esecuzione descritta può sembrare irragionevomente complicata; dopotutto il protocollo che viene usato in un portatile per leggere dal disco fisso è molto più semplice. Ad esempio usare un solo comando del tipo: **leggi questo record** invece dei tre comandi è sicuramente più facile dal punto di vista di chi deve scrivere un Device driver. In questo caso però (stiamo parlando di un sistema storage) abbiamo delle possibilità di caching delle informazioni e di virtualizzazione all'interno del sistema molto potenti: spezzare l'esecuzione nel modo descritto permette di sfruttare al meglio queste possibilità e di farle governare dal sistema operativo. Inoltre questa modalità di esecuzione è la migliore per poter gestire le situazioni in cui si verifichi un I/O error: si è in grado di dare al sistema operativo il maggior numero possibile di informazioni sulla natura dell'errore, in modo che il SW sia in grado di scegliere l'azione correttiva più appropriata.

Strumenti di misura

Nella definizione architetturale del Channel Subsystem sono incluse alcune funzioni il cui scopo è quello di fornire dati per permettere al SW di avere la misura il più esatta possibile di quanto efficaci siano le operazioni di I/O. L'insieme di queste funzioni prende il nome di "Channel Subsystem Monitoring". L'elenco che segue non è esauriente ma da un'idea dei dati ottenibili con queste funzioni:

- **Device Connect Time –** Tempo in cui il Device è attivo e sta comunicando con il canale (ad esempio sta trasferendo dati);
- **Device Disconnect Time –** Tempo in cui il Device è attivo ma non sta comunicando con il canale (ad esempio in un sottosistema storage il dato richiesto non si trova nella cache del sottosistema e si deve andarlo a prelevare fisicamente dal disco);
- **Function Pending Time –** Tempo in cui l'operazione è stata accettata dal Channel Subsystem ma non ancora accettata dal device;
- **Control Unit Queuing Time –** Tempo in cui il device è attivo, non comunica con il canale, ma è accodato in qualche funzione interna alla Control Unit. Serve a capire quanto del Device Disconnect Time è causato da problemi interni alla Control Unit;
- **Device Busy Time –** Tempo in cui il device è impegnato (ad esempio perché sta eseguendo un'operazione con un altro sistema). Fa parte del Function Pending Time.

Questi tempi vengono raccolti per ogni Subchannel per i quali sia stata abilitata la funzione di "Measurement Block Update", e letti dai SW che realizzano gli strumenti di misura. Ogni sistema operativo che opera sul mainframe ha almeno un componente che svolge questo compito.

2.0 Hardware dei Sistemi Centrali IBM (z10 EC)

Illustreremo in questo capitolo un modello recente di mainframe introdotto per mostrare come i concetti architetturali si realizzino in un'offerta per il mercato informatico. La figura 1 mostra i building blocks che compongono uno z10 Enterprise Class (EC).

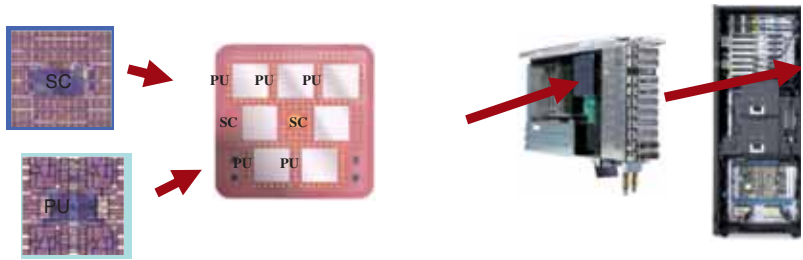


Figura 1 Building Blocks z10 EC

Da sinistra a destra abbiamo:

- I chip. Ve ne sono di due tipi:
  - Processing Unit (PU) chip (ospita i processori)
  - Storage Control / SMP Hub (SC-SMP Hub) chip (sono dei chip di interconnessione con una cache)
- Il Multichip module (MCM) si sui sono assemblati i chip
- Il book che comprende un MCM, la RAM e le connessioni per l'I/O
- La macchina completa che comprende fino a 4 book e le I/O cage (fino a 3) che ospitano gli adapter per le connessioni al network o allo storage.

2.0.0 PU chip z10

Questo è il chip che ospita i processori secondo la definizione architetturale. Esso (figura 2) contiene 4 *core*. L'implementazione dell'architettura sul modello z10 è 1 core = 1 processore. I core sono disposti simmetricamente sul chip, ciascuno circondato su tre lati dalla sua cache di livello 1.5 (3 MB). Il chip include anche un *memory controller*, un *I/O bus controller* e uno *switch* che connette tutti e quattro i core a un'interfaccia condivisa con l'SMP hub chip e la sua cache. Il processore z10 è implementato nella tecnologia IBM *silicon-on-insulator* (SOI) a 65nm, la dimensione del *dye* è di 454 mm^2, contiene 994 milioni di transistor e opera a 4.4 GHz in un sistema multiprocessore.
Il core z10 implementa la z/Architecture. Per lo z10 IBM ha aggiunto più di 50 istruzioni alla z/Architecture, per migliorare il throughput del processore. Il core z10

aggiunge inoltre il supporto per *page* (cioè pagine virtuali) da 1MB e interfacce software-hardware per migliorare l'efficienza della cache.
Il core del microprocessore z10 è organizzato in 8 unità come mostrato nella Figura 3. La *instruction fetch unit* (IFU) contiene una cache di 64 Kbyte per le istruzioni, la logica di *branch prediction*, i controlli per *l'instruction-fetching*, e gli *instruction buffer*. Gli *instruction buffer* nella IFU alimentano la *instruction decode unit* (IDU) al centro del core. La logica analizza e decodifica i quasi 900 distinti *opcode* definiti nella z/Architecture, identifica le dipendenze tra istruzioni, forma le coppie di istruzioni per l'esecuzione superscalare quando possibile, e immette le istruzioni nelle parti di logica di accesso agli operandi e di esecuzione.
Le istruzioni dell'architettura sono implementate in HW (la maggior parte) o in Millicode (quelle più complesse).

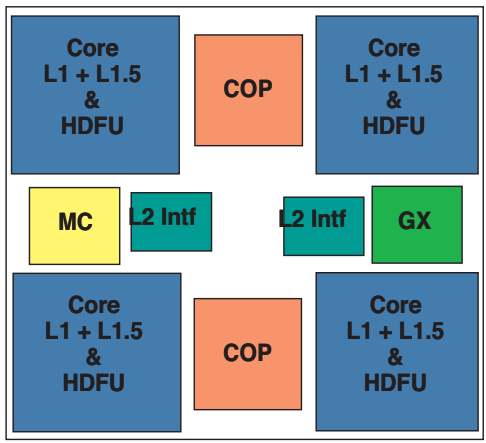


Figura 2 Chip del processore z10

La *load-store unit* (LSU) include una cache di 128-Kbyte per i dati e gestisce gli accessi agli operandi trattando tutte le lunghezze, modi e formati previsti nella z/Architecture, e supporta due *fetch* di blocchi di 128 bit (*quadword*) per ciclo. Inoltre essa bufferizza i risultati di operazioni *(operand store)* tra l'esecuzione dell'istruzione e il suo completamento, e si interfaccia con la *fabric* multiprocessore (attraverso la cache di livello 1.5) per mantenere la coerenza fortemente ordinata della cache richiesta dall'architettura. La LSU è accoppiata con una *translation unit* (XU), che è costituita da un ampio *translation look-aside buffer* (TLB) di secondo livello e un'unità di traduzione hardware. Quest'ultima gestisce la traduzione degli Access Register e la traduzione dinamica degli indirizzi (*dynamic address translation* o DAT) per convertire indirizzi virtuali in indirizzi reali, inclusa la *nested* DAT richiesta per i sistemi operativi in esecuzione come ospiti sotto l'ipervisore z/VM (macchine virtuali).

Tre unità gestiscono l'effettiva esecuzione delle istruzioni: la *fixed-point unit* (FXU) esegue istruzioni aritmetiche in virgola fissa, logiche e di branch. La FXU esegue la

maggior parte di queste in un solo ciclo, e in coppie, con una completa rete di *forwarding* per permettere l'esecuzione di operazioni tra loro dipendenti l'una di seguito all'altra. La *binary floating-point unit* (BFU) è una *pipeline* multistadio che gestisce tutte le operazioni in virgola mobile binarie (secondo lo standard IEEE-754) ed esadecimali (eredità dell'architettura S/360). Questa unità può avviare un'operazione per ciclo, e utilizza il *forwarding* intrapipeline dei risultati per minimizzare nella pipeline le latenze tra istruzioni dipendenti. La HW *decimal floating point unit* (DFU) esegue le operazioni decimali in virgola mobile (standard IEEE-754R). Questa è un'importante innovazione del processore z10. In precedenza, fino allo z9, queste funzioni venivano svolte in millicode o dal software. Con l'aggiunta del HDFU, le operazioni svolte direttamente dal core migliorano le prestazioni fino ad un massimo di 10 volte. Inoltre vengono evitati tutti gli arrotondamenti ed altri problemi dovuti alla conversione da binario a decimale. Questa novità è estremamente importante per tutte le applicazioni finanziarie e commerciali.

In ultimo, la *recovery unit* (RU) mantiene una copia completa dello stato architetturale del processore, protetta da un codice di correzione d'errore (error-correction code o ECC). Questo stato include sia tutti i registri previsti dalla z/Architecture sia i vari registri di modo e stato usati dall'hardware e dal millicode per implementare le funzioni della z/Architecture. La RU raccoglie tutti i segnali hardware di *fault detection* e sovraintende alle azioni hardware di recupero se questi segnali indicano una malfunzione.

La caratteristica che più di tutte contraddistingue il *design* del core del microprocessore z10 rispetto ai predecessori è il gigantesco salto nella frequenza di esercizio - da 1.7 GHz sul sistema z9 a 4.4 GHz sui sistemi basati su z10. A partire dal modello CMOS G4 uscito nel 1997 i chip dei mainframe IBM hanno sempre avuto un ciclo della durata da 27 a 29 FO4[1] e una pipeline lunga 6 cicli, contando dalla fase di decodifica delle istruzioni alla fase di scrittura dei registri (*put-away*). Attraverso sei generazioni di sistemi e tecnologia elettronica, i progettisti hanno mantenuto costante la dimensione del ciclo e la profondità della pipeline mentre hanno aggiunto funzioni di tutto rilievo (come la capacità di esecuzione di operazioni in virgola mobile in conformità con lo standard IEEE, la *branch target prediction*, la completa estensione architetturale a 64 bit, la modalità di esecuzione superscalare e la crittografia).

[1] Il FO4 è il ritardo introdotto da un inverter con fanout 4. L'unità è usata dagli specialisti del settore per confrontare tempi in maniera indipendente dalla specifica tecnologia di fabbricazione.

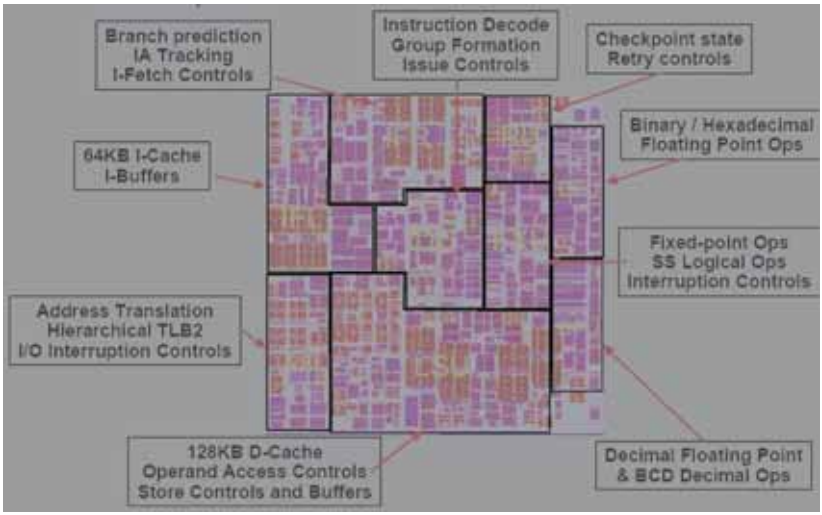


Figura 3 Le unità del core del microprocessore z10

Il progetto del chip z10, tuttavia, è partito da un foglio bianco, con l'obiettivo di ottenere un ciclo molto più corto (15 FO4), e di bilanciare prestazioni, potenza e area occupata dal chip nel quadro di una notevole complessità progettuale. Questo cambiamento ha richiesto innovazioni nella metodologia di progetto, nella struttura della pipeline e nell'implementazione dell'architettura (vedi figura 4).

Sul chip sono presenti due coprocessori esterni ai 4 core ognuno dei quali implementa funzioni crittografiche e di compressione dei dati; ogni coprocessore è usato da due dei core presenti sul Chip (questa condivisione può essere fonte di contesa fra i due core e viene monitorata dalla strumentazione HW della macchina). Queste funzioni sono previste dalla z/Architecture, e sono accessibili dal SW come istruzioni sincrone convenzionali (queste istruzioni sono state implementate anche usando il millicode).

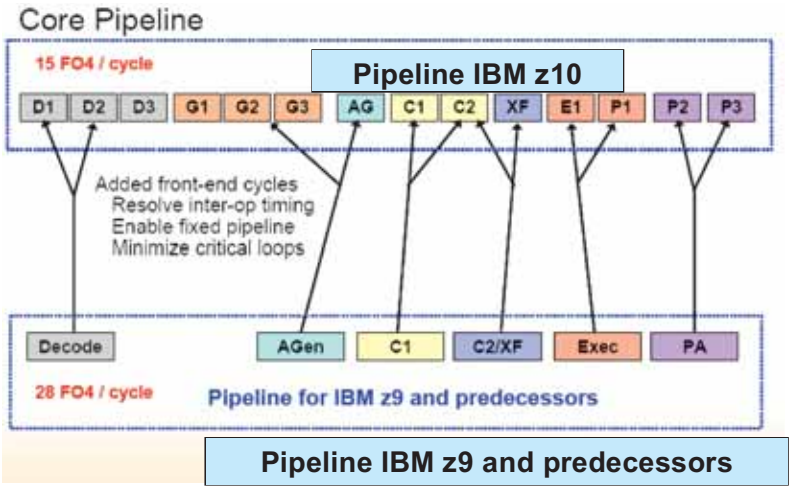


Figura 4 Confronto tra le pipeline del processore z10 e quelle dei suoi precedecessori

Ogni coprocessore contiene due motori di compressione (ognuno con una cache locale di 16 Kbyte), un motore di cifratura crittografico e un motore per il calcolo dello hash crittografico.

Per concludere si può osservare che il modello z10 ha un core, un chip e un disegno multiprocessore unici nel loro genere e sviluppati specificatamente per il mercato dei *data server* di fascia enterprise, anche se dal punto di vista tecnologico condivide le tecniche di progettazione ad alta frequenza e i blocchi elettronici di base con il processore Power6, utilizzato nei sistemi IBM che implementano l'architettura RISC Power.

Il core z10 a 4.4 GHz rende fino al doppio in prestazioni rispetto alla generazione precedente z9 per applicazioni che fanno un uso intensivo della CPU (*CPU-intensive*) e che costituisono una parte crescente dei carichi di elaborazione dati delle aziende. In coppia con la cache privata di 3 Mbyte (Livello 1.5) private e la cache condivisa di 48 Mbyte (Livello 2), il design dello z10 fornisce un guadagno del 50 per cento su un'ampia gamma di transazioni tradizionali che fanno un uso intensivo del sottosistema di I/O (*I/O-intensive*) e che continuano a costituire la maggior parte dei carichi di lavoro commerciali. Questo design fornisce la base per estendere la piattaforma mainframe IBM con l'evolversi della tecnologia elettronica su silicio per molte future generazioni.

Il Millicode

L'instruction set della z/Architecture diventa sempre più ricco: istruzioni e funzioni sempre più complesse sono state introdotte. Concettualmente le istruzioni più semplici (LOAD, STORE, MOVE, BRANCH) e le istruzioni logiche e aritmetiche possono essere implementate direttamente nell'hardware. Le istruzioni e le funzioni più complesse come le istruzioni di I/O, la Start Interpretive Execution (SIE), le istruzioni cross-memory, i gestori delle interruzioni e certe caratteristiche RAS, devono essere implementate con qualche tipo di codice interno. A partire dal processore S/390 G4 nel 1997 fino all'attuale z10 il codice interno al processore è chiamato millicode.

Su molti processori precedenti al G4, il codice interno fu posto su chip separati; questo è noto come microcode orizzontale. Con il sistema G4, l'intero processore fu implementato su un unico chip. A causa dei vincoli di area del chip, si impose un redesign del codice interno del processore, dal momento che esso non avrebbe più potuto essere contenuto nel chip, e, con l'aggiunta di nuovi requisiti a livello architetturale, sarebbe stata necessario una maggiore quantità di codice interno. Questo portò al progetto del millicode verticale come codice interno del processore. Il millicode è scritto in linguaggio assembler, in primis con istruzioni della z/Architecture implementate nell' Hardware, in secundis con istruzioni specializzate per il solo millicode.

Il Millicode è un mezzo molto potente per aggiungere funzioni sofisticate all'architettura senza appesantire inutilmente l'essenza HW dei Chip z.

2.0.1 SC-SMP Hub Chip z10

Questo è un chip che ha varie funzioni. E' progettato specificatamente per l'utilizzo con i processori z10 e permette la costruzione di sistemi SMP altamente scalabili. Il chip SC-SMP Hub connette vari PU chip z10, permettendo di raggiungere una banda passante di 48 Gbytes/s tra SC-SMP Hub chip e PU chip. Il chip SC-SMP Hub include una cache SRAM (Static Random Access Memory) da 24 Mbyte; coppie di chip SC-SMP Hub sono connesse per realizzare una cache condivisa (Livello 2) unica di 48 Mbyte per l'insieme dei PU chip presenti sul Multi Chip Module. Fino a 4 coppie di chip SC-SMP Hub possono essere collegate per formare sistemi SMP più grandi; la *fabric* a bassa latenza permette di scalare efficientemente in senso SMP e supporta i requisiti di coerenza forte della z/Architecture.

Questo struttura con switch centrale e cache condivisa rispetto a topologie di *fabric* più distribuite procura notevoli vantaggi in termini di prestazioni per carichi di lavoro con un alto grado di condivisione dati e di interazione tra processi, che sono comuni in molti ambienti di elaborazione di transazioni commerciali. Implementato a livello di processo di fabbricazione nella stessa tecnologia del PU chip z10, questo chip è costituito da 1.6 miliardi di transistors su un dye di 445-mm^2 .(vedi figura 5).

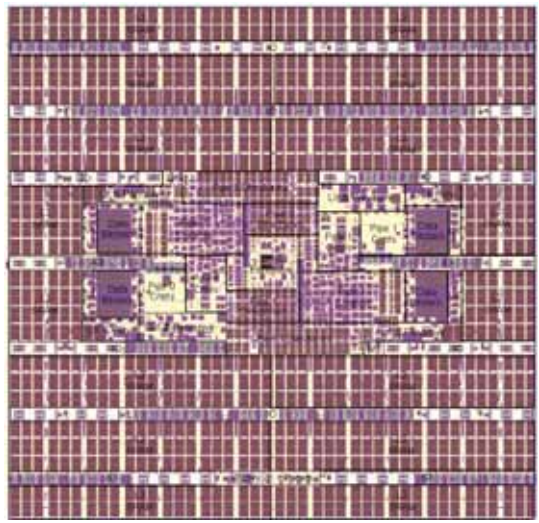


Figura 5 Il chip SC-SMP Hub

Maggiori dettagli sul processore z10 e il chip SC si possono trovare nell'articolo di Charles F. Webb **IBM z10: The Next-Generation Mainframe Microprocessor**[2]

2.0.2 z10 Multiple Chip Module (MCM)

Sul Multi Chip Module (MCM) z10 alloggiati 7 chip. Cinque sono PU chip e due sono SC-SMP Hub chip. Questi sette chip svolgono tutte le funzioni necessarie, e cioè processori (PU), le memorie cache, system data cache (SD) e storage control (SC), funzioni di controllo d'accesso alla memoria, Memory Subsystem Control (MSC) ed il clock.
Le dimensioni del modulo MCM sono paragonabili a quelle di un comune floppy disk, 95x95mm, ed è fatto in ceramica sulla quale sono montati i chip composti da circa 8 miliardi di transistors (vedi figura 6).

[2] L'articolo è pubblicato sul numero 2 della rivista IEEE Micro, volume 28, marzo-aprile 2008.

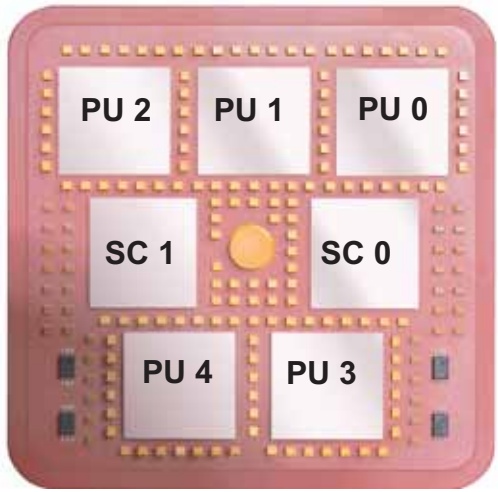


Figura 6 z10 MCM

La tecnologia utilizzata da tutti i chip è la CMOS 11S, basata su 10 strati di interconnessione in rame ed isolamento in Silicio. Lo spessore litografico è di 0,065 micron. Come detto, i 5 chip "quad-core" forniscono 17 o 20 core utilizzabili. Nella versione a "17", 3 chip forniscono 3 soli processori (3x3 = 9), e gli altri 2 ne forniscono 4 (2x4 = 8) per un totale di 17. Nella versione a "20" tutti e 5 i chip forniscono 4 processori (5x4 = 20). Il ciclo base è per tutti i chip è di 0,23ns.

In un MCM a "17" , la seguente tabella illustra lo scopo a cui vengono destinati i processori contenuti nei chip "quad-core":

Modello	PU applicativi	SAP	Spare	Totale
E12	12	3	2	17
E26	26	6	2	34
E40	40	9	2	51
E56	56	10	2	68

Il modello di fascia alta, z10 EC E64, usa un MCM da "17" e tre MCM da "20" i cui processori vengono usati in questo modo:

Modello	PU applicativi	SAP	Spare	Totale
E64	64	11	2	77

I processori System Assist Processor (SAP) sono assegnati al Channel Subsystem per svolgere operazioni di I/O e i processori Spare svolgono la funzione di processori di "scorta". Analogamente all'elaboratore z9, anche lo z10 EC, qualunque sia il numero di MCM, ha al massimo solo 2 processori di scorta.

Come visto in precedenza, ogni core ha a disposizione, sul proprio chip, una cache di L1 suddivisa in 64KB di cache per le istruzioni e 128KB per i dati e una cache intermedia, L1,5, di 3MB per processore.

La funzione di questa cache è di ospitare il contenuto della L1 che deve essere spostato per far posto a nuovi dati secondo il concetto del Least recently used (LRU).

La cache di L1 è la cache da cui vengono prelevati instruzioni e operandi necessari per l'esecuzione dei programmi. Se il dato non si trova in questa cache esso deve essere prelevato dai livelli di memoria a valle, portato in L1 e quindi fornito al processore.

Lo Storage Control chip (SC-SMP Hub) controlla e gestisce il traffico tra PU e L2, tra L2 e le altre L2, tra L2 gli altri componenti, MSC e MBA.

La cache L2 è alloggiata sui 2 chip SC-SMP Hub, ognuno con una capacità di 24MB; i due set da 24MB costituiscono un unico spazio indirizzo da 48 MB.

L'algoritmo di scrittura è Store Through fra L1, L1.5 e L2; ogni scrittura su L1 viene replicata in modo sincrono - rispetto all'esecuzione dell'istruzione – su L1.5 e L2. E' Store In fra L2 e la memoria reale (le scritture su L2 non sono replicate in modo sincrono sulla memoria reale).

2.0.3 z10 Book

L'altro elemento fondamentale della struttura dei sistemi z10 è il Book (libro), dove viene ospitato il MultiChip Module (MCM). I sistemi z10 usano la stessa tecnica di packaging usato per la famiglia precedente, z9, basato sul concetto di Book. Un Book contiene i processori, alloggiati all'interno del MCM, la memoria centrale, una combinazione di Memory Bus Adapter (MBA) e Host Channel Adapter per la connessione verso i dispositivi di I/O (vedi figure 7 e 8).

I sistemi z10 EC hanno da 1 a 4 Book. Tutti i Book risiedono all'interno della CEC (Central Electronic Complex) cage dell'elaboratore.

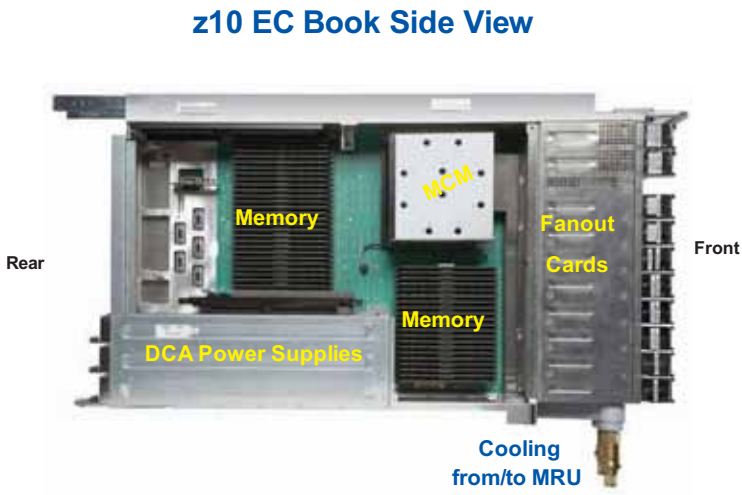


Figura 7 Vista laterale di un book z10

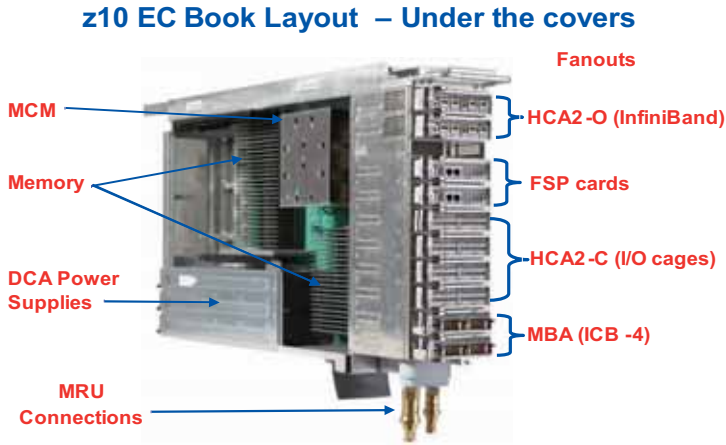


Figura 8 Dettagli di un book z10

In dettaglio, gli elementi contenuti all'interno di un singolo Book sono i seguenti:

- un MCM da 17 o 20 processori (PU)
- memoria da 64 a 384GB
- fino a 8 Fanout Card, HCA o MBA, per la connessione verso le I/O
- tre Distributed Converter Assemblies (DCA), che forniscono "power" al Book

Il raffreddamento dell'elaboratore z10 EC è di tipo ibrido, in quanto ci sono due sistemi indipendenti, uno ad aria, "air-cooled system", ed uno ad acqua, ottenuta grazie ad un sistema interno a liquido refrigerante.

I Book di cui è composto un sistema z10 sono interconnessi tra loro per permettere la comunicazione e la condivisione delle risorse di memoria e di I/O tra tutti i processori presenti nell'elaboratore. Dal punto di vista della gestione Hardware esiste un unico Address Space per la memoria Hardware, che comprende tutta la memoria installata su tutti i libri presenti nella configurazione. Logicamente la memoria, che dal punto di vista costruttivo è allocata su Book diversi, è vista come un unico spazio indirizzabile che può arrivare fino a 1,5TB.

Un punto essenziale della struttura a libri è che le cache di L2 memorizzano i dati acceduti dai CP presenti sul libro stesso. Non sono legate alla memoria installata sul Book. Una conseguenza di questo disegno è che una certa posizione di memoria può essere presente in più di una cache L2. Questo principio costruttivo è essenziale per riuscire a mantenere un Cache Hit elevato.

E' abbastanza evidente che per mantenere un elevato livello di prestazioni è necessario che ogni CP acceda il più possibile a dati che sono presenti sul suo Book o perché sono presenti sulla memoria del Book, o perché si trovano sulla cache di L2 del proprio Book. Per ottenere questo si è resa necessaria una profonda revisione degli algoritmi che regolano il funzionamento del Partizionamento logico e delle funzioni di Dispatching. Per un approfondimento su questo tema si rimanda al seguente articolo di "IBM Journal of Research and Development"
http://www.research.ibm.com/journal/rd/483/siegel.html

La connessione tra i diversi Book è di tipo point-to-point, che consente ad ogni Book di comunicare con tutti gli altri in maniera diretta (vedi figura 9).

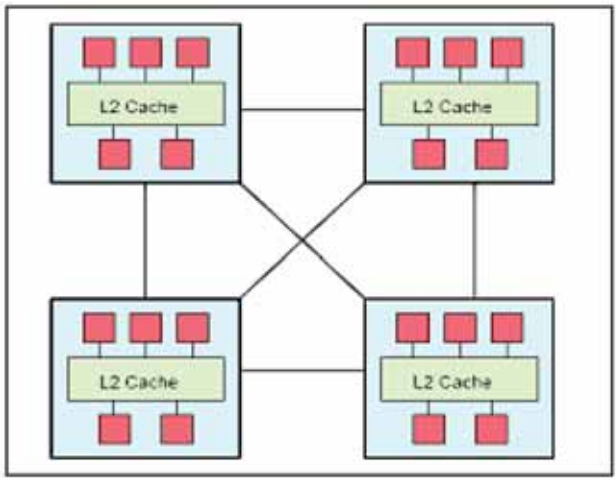


Figura 9 Comunicazione tra i Book

La connettività verso i dispositivi di I/O è assicurata da una combinazione di otto adapters di tipo Memory Bus adapter (MBA) o Host Communication Adapter (HCA) per book, localizzati nella parte frontale. Ogni adapter HCA ha due connessioni di tipo InfiniBand. Ogni connessione ha una portata di 6 GB/Sec. Si usano due diversi tipi di HCA:

- HCA2-O: connessione ottica di tipo Coupling Link per realizzare il parallel SYSPLEX;
- HCA2-C: connessione in rame per connettere il book alle I/O Cage che ospitano le blade di I/O (vedere più avanti).

Gli MBA sono usati solo per connettere lo z10 a macchine z di generazione precedente; queste connessione possono essere necessarie solo quando si realizza il parallel SYSPLEX: perciò gli MBA sono praticamente assenti sugli z10 attualmente installati.

2.0.4 La specializzazione dei processori

Un MCM contiene 17 oppure 20 processori (core). A questi processori viene assegnata una "caratterizzazione", ovvero un ruolo, mediante un opportuno firmware[3] caricato all'accensione della macchina. Una volta caratterizzati, i processori possono essere di due tipi, Utente e Servizio.

[3] programma nativo del computer e non modificabile dall'utente, punto di incontro fra componenti logiche e fisiche, ossia fra hardware e software

I processori Utente sono:

- **Central Processor (CP):** possono essere utilizzati da qualunque sistema operativo; sono anche chiamati Processori Standard o General Purpose.
- **Integrated Facility for Linux (IFL):** usati esclusivamente da sistemi per Linux e z/VM.
- **zSeries Application Assist Processor (zAAP):** dedicati all'esecuzione di applicazioni Java che girano sotto z/OS.
- **zSeries Information Assist Processor (zIIP):** dedicati all'esecuzione di attività legate al DB2 e alla Crittografia (IPSEC) sotto z/OS.
- **Integrated Coupling Facility (ICF):** usati nella realizzazione della struttura di Cluster di Sistemi z/OS denominata Parallel Sysplex.

I processori di Servizio sono:

- **Service Access Processor (SAP):** legati alla gestione delle Operazioni di I/O. Forniscono la potenza di calcolo per realizzare la funzione architetturale *Channel Subsystem.*
- **Spare Processor (Spare):** presenti sull'MCM ma non attivi, essi vengono attivati automaticamente in caso di guasto di uno degli altri sia di Servizio che Utente; in un Sistema z10 EC sono sempre presenti almeno due processori Spare.

Tutti questi tipi sono assolutamente identici dal punto di vista HW. La loro caratterizzazione è esclusivamente basata sul fatto di poter fare o no certe funzioni. La caratterizzazione di un CP è gestita con opportune modifiche alla configurazione logica della macchina senza alcun intervento HW.

I processori Utente denominati zAAP e zIIP sono stati introdotti per ottimizzare l'infrastruttura IT che ospita l'Application Server e il Database Server. Fino a qualche anno fa, una delle soluzioni più adottate era quella di realizzare reti di serventi inteconnesse tra loro, distribuendo le funzioni da svolgere tra i vari componenti. Ad esempio, alcuni serventi eseguono i programmi applicativi ("application server"), altri accedono ai dati ("database server"). Una rappresentazione schematica è quella illustrata nella configurazione di sinistra di Figura 10.

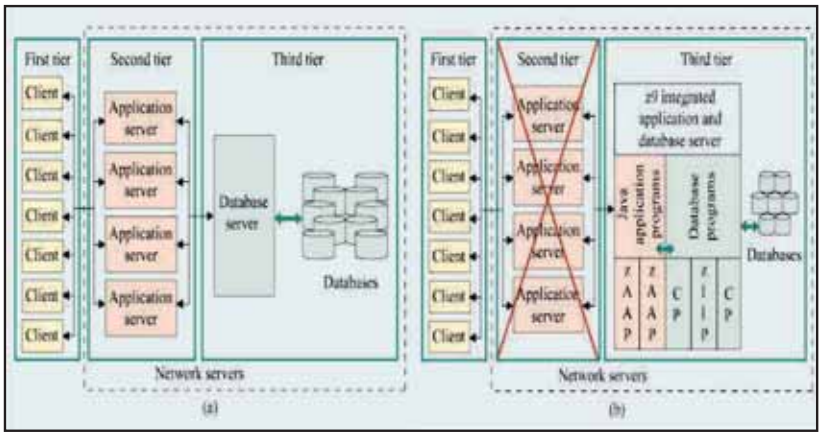


Figura 10 Multitiered networks (a) distributed (b) simplified with zIIP+zAAP

Con il tempo, il numero di serventi applicativi è cresciuto in maniera significativa, specialmente per quelle applicazioni disegnate per supportare i nuovi modelli architetturali internet-oriented o web-based. All'interno di questi modelli emergenti le nuove tecnologie di programmazione Java e XML[4] giocano un ruolo fondamentale, in quanto offrono dei vantaggi dal punto di vista della produttività nello sviluppo dei programmi, riducendone i tempi ed i costi associati.
Conseguenze non volute di questa proliferazione dei serventi distribuiti ("server farm") sono la riduzione della affidabilità totale dell'infrastruttura informatica, l'aumento della complessità di gestione e l'inefficienza nell'utilizzo dei serventi. Per risolvere questi problemi si è provveduto ad ospitare all'interno dei sistemi z/OS sia gli Application Server che i Database Server. I processori zAAP sono delegati all'esecuzione degli Application Server basati su Java, i processori zIIP supportano i Database Server basati su DB2. Il processore zAAP esegue istruzioni di codice applicativo Java; sul processore zIIP vengono svolte svariate funzioni:

- parte del carico generato dalle funzioni di gestione del database relazionale DB2 per z/OS
- L'accesso ai dati Db2 provenienti da applicazioni esterne (che cioè girano in partizioni logiche differenti o su altre piattaforma). In termini SW sono accessi generati dal protocollo Distrubuted Relational Database Architecture (DRDA) quando si usa il protocollo TCP/IP.
- Operazioni relative alla sicurezza IP.

Insieme, questi processori riescono ad eseguire una considerevole quantità del carico di lavoro richiesto dalle suddette funzioni. Di conseguenza, liberano della capacità di

[4] Extensible Markup Language

calcolo a carico dei processori cosiddetti "general purpose" che la rendono disponibile ad altri utenti. In questo modo, la piattaforma IBM System z10 è in grado di offrire ottime prestazioni, funzioni avanzate ed un costo estremamente competitivo in relazione a soluzioni distribuite più complesse e meno efficienti. La configurazione di destra della Figura 10 illustra una soluzione che usa motori Standard, zAAP e zIIP con una evidente semplificazione di tutta l'infrastruttura. Entrambi i processori operano in maniera completamente trasparente ai programmi Java che usano lo zAAP e alle funzioni di database DB2 che usano lo zIIP. Il sistema operativo z/OS e il meccanismo di Partizionamento logico coordinano l'attività di motori Standard, zAAP e zIIP. Tutto questo viene svolto in modo trasparente agli Application Server e Database Server.

A differenza dei processori standard (CP), zAAP e zIIP hanno delle caratteristiche particolari quali, ad esempio, l'impossibiltà di eseguire la funzione di IPL[5], cioè il caricamento iniziale del sistema operativo o, eventualmente, di un programma di utilità particolare, o di eseguire istruzioni legate alle operazioni di I/O.

2.0.5 Il Channel Subsystem z10

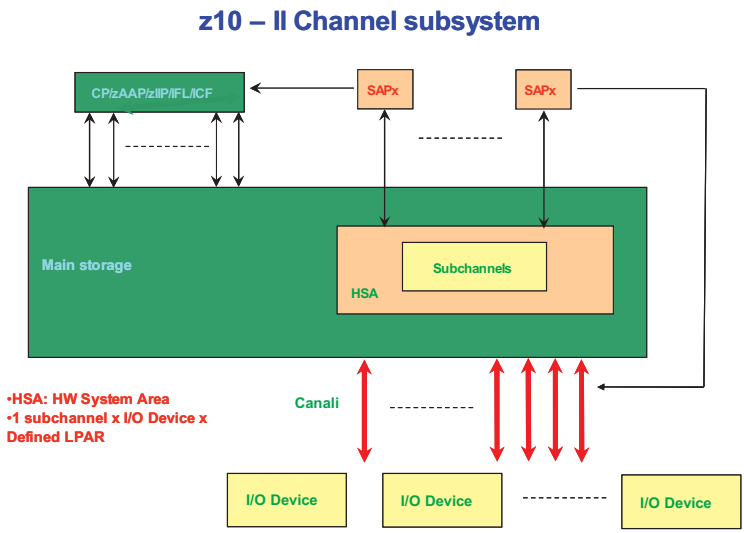


Figura 11 Il Channel Subsystem z10

Facendo riferimento alla figura 11, le risorse HW che il Channel susbsystem ha a disposizione su uno z10 sono: i processori SAP, una parte di memoria chiamata HW

[5] Initial Program Load

System Area (HSA), i canali. Il Channel Subsystem ha un suo sistema operativo di architettura z che svolge tutte le funzioni architetturali di un Channel Subsystem. L'HSA non e' accessibile dagli altri CP, ma il suo contenuto può essere interrogato dai sistemi operativi con delle funzioni definite dall'architettura (ad esempio Channel Subsystem Call). Il numero di processori SAP cresce con il numero di book installati. Si va da un minimo di 3 SAP a un massimo di 11, secondo il principio che, maggiore è il numero di CP, maggiore è il numero di operazioni di I/O al secondo che si generano.
Il contenuto della HSA è costituito da:

- Il codice che implementa le funzioni del Channel subsystem
- Una serie di blocchi di controllo che definiscono la configurazione accessibile al server. La parte preponderante di questi blocchi di controllo è costituita dai subchannel. Sono aree assegnate ad ogni I/O device definito per ogni Partizione Logica definita. Perciò se la configurazione prevede 30000 I/O device e sono state definite 20 partizioni logiche, si avranno 600000 subchannel.

I CP (Standard, IFL) eseguono l'istruzione Start Subchannel. L'effetto di questa istruzione è di notificare al Channel subsystem che c'è del nuovo lavoro da fare. Si noti che eseguire l'istruzione di start subchannel implica, fra l'altro, che, dal punto di vista della Partizione Logica non c'e' coda per l'accesso al Device.

Il Channel subsystem verifica che il device sia libero (potrebbe essere usato da altre partizioni logiche sullo stesso server) e, in caso positivo inizia l'operazione di I/O come è stato descritto nel capitolo sull'architettura. I SAP lanciano l'attività sui canali i quali provvedono autonomamente a dialogare con i device secondo lo schema illustrato nel capitolo dell'architettura. I canali hanno un accesso diretto alla memoria (Direct memory access – DMA) cioè non impegnano i SAP se non per iniziare e concludere l'operazione.

Il Channel Subsystem usa i Canali per raggiungere i device: è quindi essenziale che abbia una descrizione assolutamente esatta della configurazione di I/O. Questa descrizione si trova memorizzata in file particolari che si chiamano I/O Configuration Data set (IOCDS). Questi file sono memorizzati non sul mainframe ma su un PC di servizio chiamato Support Element (SE) che verrà illustrato più avanti.

Questa definizione si basa su un SW chiamato I/O Configuration Program (IOCP).

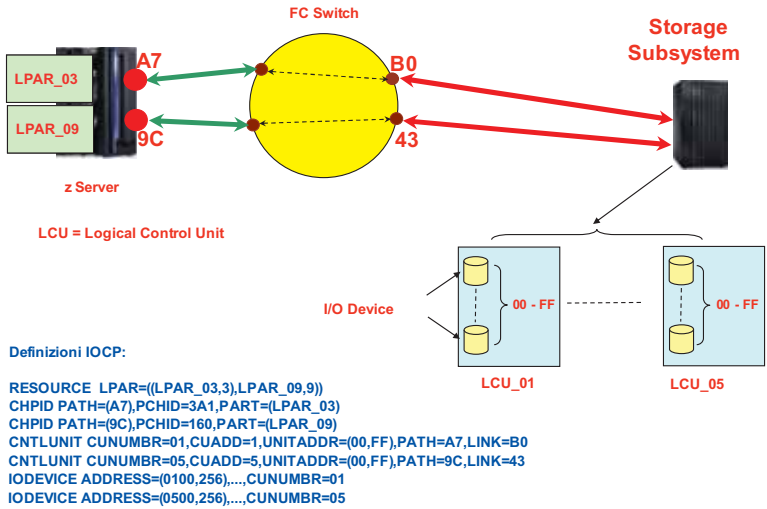


Figura 12 La definizione IOCP

La figura 12 illustra i principi fondamentali di questa definizione. Nell'esempio abbiamo:

- Un server con due partizioni logiche (LPAR_03 e LPAR_09)
- Due canali (CHPID: A7 e 9C) per connettere uno storage subsystem
- Uno switch Fiber Channel
- Uno storage susbsytem. In fase di set up questo unico box è stato diviso in due Control Unit logiche (LCU_01 e LCU_05). La definizione architetturale di Control Unit si applica alle Logical Control Unit, non allo storage subsystem.
- Ogni logical control unit accetta operazioni per device che abbiano uno Unit Address compreso fra x'00' e x'FF'.
- Ognuna di queste due Control Unit controlla 256 volumi di dati.
- Ogni volume costituisce un I/O device.

Gli statement IOCP necessari per descrivere questa configurazione e trasformarla in un IOCDS sono:

- **RESOURCE -** Identificano le partizioni che useranno la configurazione. Nell'esempio ne abbiamo due: LPAR_03 e LPAR_09. I valori 3 e 9 che sono associati ai nomi delle partizioni verranno inclusi in tutte le frame Fiber Channel durante l'esecuzione delle operazioni e prendono il nome di Multi Image Facility Identifier (MIF Id). E' necessario poichè un canale FC può essere usato da più di una partizione logica. Per istradare correttamente tutte le frame che realizzano l'operazione di I/O ognuna di queste frame deve avere un'indicazione di quale partizione ha richiesto l'operazione. Questa indicazione è il MIF-Id.
- **CHPID -** Indicano i canali che la configurazione ha a disposizione. **PCHID** è l'indicazione della posizione fisica dell'adapter all'interno del server. **PART**

indica quali partizioni possono usare questo canale. Nell'esempio si è deciso che il canale A7 sia usato solo dalla partizione LPAR_03, e il canale 9C solo dalla partizione LPAR_09

- **CNTLUNIT -** Definiscono le control unit presenti nella configurazione. Abbiamo uno statement per ogni Logical Control Unit. **CUNUMBR** assegna un'identificazione alla Control Unit che vale all'interno di questa definizione. UNITADD indica il range di Unit Address che questa Control Unit gestisce. La prima Control Unit sarà acceduta dal CHPID A7, la seconda dal CHPID 9C. **CUADD** è l'identificatore all'interno dello storage subsystem relativo alla Control Unit Logica. **LINK** indica la porta sullo switch a cui il channel adapter sullo storage susbsystem è collegato. L'insieme di PATH e LINK definiscono una connessione all'interno dello switch tutte le volte che l'operazione di I/O dovrà, transitando attraverso lo switch arrivare al device o tornare al canale.
- **IODEVICE –** Sono tutti gli I/O device presenti nella configurazione. **ADDRESS** è il device number assegnato al device. Poichè non si è indicato sullo statement IODEVICE la keyword UNITADD si assume che gli unit address associati ai vari device siano uguali agli ultimi due digit esadecimali del device number. Ad esempio il device con device number 013B avrà come Unit Address x'3B'.

L'esempio non esaurisce tutte le possibilità che l'IOCP mette a disposizione per descrivere le configurazioni di I/O che sono di fatto molto più complicate che nell'esempio.
Si noti inoltre che con questa definizione la control unit LCU_01 (con i device a valle) sarà accessibile solo da LPAR_03, mentre LCU_05 verrà acceduta solo da LPAR_09 (poichè si è stabilito che A7 e 9C sono acceduti da una sola partizione).
Se questa è la definizione come fanno il channel subsystem, i canali e le control unit a raggiungere i device per effettuare le operazioni di I/O?
Tutta la definizione che abbiamo visto dà come risultato principale la creazione di 1024 Subchannel per descrivere i device: 256 device per control unit acceduti da due partizioni. Questi subchannel alla partenza del server (fase che viene indicata con Power On reset – POR) vengono creati e posizionati in HSA.
In ogni subchannel vengono messe, fra le altre, queste informazioni: CHPID che permettono di arrivare al device, MIF-ID della partizione logica associata al subchannel, CUADD relativo alla Logical Control Unit, Unit Address da usare per indirizzare il device all'interno della Logical Control Unit. CUADD, Unit Address e MIF-Id sono inseriti in tutte la frame relative a operazioni sul device. A ognuno di questi subchannel viene associato un numero unico nell'ambito del channel subsystem. Il sistema operativo che parte in una partizione (ad esempio in LPAR_09) per conoscere la configurazione che ha a disposizione interroga il Channel subsystem facendosi dare il contenuto di tutti i subchannel che è abilitato ad usare. Nel nostro esempio sono i subchannel associati a quei device con Device Number da 0500 a o5FF.
Ottenuta questa informazione il sistema operativo provvede a costruire la versione SW della configurazione, che comprende dei blocchi di controllo per i device che può usare. Ad esempio in z/OS questi blocchi di controllo si chiamano Unit Control Block (UCB). In ogni UCB viene scritto sia il Device Number che il Subchannel number. La figura 13 illustra questa relazione.

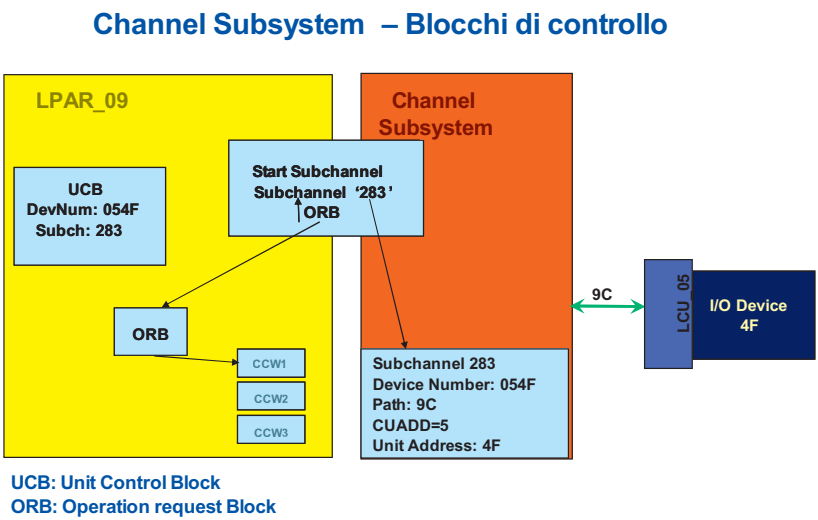


Figura 13 Flusso delle informazioni per le operazioni di I/O

Supponiamo adesso che nella partizione LPAR_09 (vedi figura 13) un'applicazione voglia accedere a dati che si trovano sul device 054F. Quando la partizione è partita ha costruito una serie di UCB; quello relativo al device 054F dice che il subchannel relativo è il 283. A questo punto il sistema operativo costruisce un Operation Request Block (ORB) che indica il programma di canale da eseguire su quale subchannel number e lancia la Start Subchannel i cui due parametri sono il subchannel e l'indirizzo dell'ORB in memoria virtuale. Il Channel subsystem entra in azione e, vedendo che il device legato al subchannel 283 ha come via di accesso il CHPID 9C, dice a questo canale di eseguire un'operazione di I/O il cui programma è indicato dall'ORB, e le informazioni sul device si trovano nel subchannel 283.

Nella sezione dedicata allo z/OS vedremo come si arriva dalla richiesta applicativa alla Start Subchannel.

2.0.6 Gli Adapter di I/O z10

Sullo z10 si possono avere le seguenti interfacce per collegare il server ai sistemi storage o al network:

- **ESCON:** connessione a fibra ottica con protocollo IBM proprietario, usata per collegare dischi, nastri, terminali di vecchio tipo e stampanti. Velocità max 17MB/sec. Ogni scheda ESCON contiene 15 canali
- **Ficon/FCP:** connessione a fibra ottica con protocollo IBM proprietario nel caso di Ficon o protocollo SCSI over FC (FCP). Usata per collegare dischi, nastri e Storage Area Network (SAN). Velocità 2, 4, 8 Gbit/sec con

connessioni laser shortwave (SX) o laser longwave (LX). Ogni scheda FICON contiene 4 canali

- **OSA Express3:** connessione standard per LAN Ethernet, supporta collegamenti in fibra ottica, SX o LX, e in rame RJ45. Usata dai due protocolli di rete in uso sulla piattaforma Mainframe TCP/IP (Standard de jure) o SNA (proprietario). Velocità da 10 a 1000 Mbit/sec, in rame, e da 1 a 10Gbit/sec in fibra. Il numero di canali è variabile. La scheda contiene due canali se è del tipo 10 Gbit Ethernet, 4 canali se è del tipo Fast Ethernet (rame) o 1 Gbit ethernet.

La tecnologia ESCON è in fase di rapido abbandono. Confrontando le caratteristiche ESCON/FC è facile rendersi conto dell'enorme vantaggio che il FC offre. Infatti lo z10 EC sarà probabilmente l'ultima generazione a offrire la possibilitò di avere fino a 1024 canali. L'uso sempre più esteso del FC fà sì che pochi adapter FC facciano il lavoro di tanti ESCON (rapporto 1 a 10).

Per le connessioni relative al Parallel SYSPLEX si usano due altri tipi di connessioni:

- **Inter System Coupling (ISC) links:** connesioni in fibra ottica per connettere LPAR fra di loro. Operano a 2 Gbit/Sec con un protocollo proprietario. Ogni scheda ISC ha 4 canali

- **Infiniband Coupling Link:** hanno la stessa funzione degli ISC ma usano lo standard Infiniband. Sono diponibili in due versioni: **1x** a 500 Mbyte/Sec con una lunghezza massima (con segnale ottico non rinforzato) di 10 Km, e **12x** a 6 Gbyte/sec con una lunghezza massima di 150 m. Ogni Adapter Infiniband può essere configurato con un massimo di 16 canali di tipo Coupling Link (anche se fisicamente si hanno solo due cavi per adapter).

Tutti questi adapter sono complessi HW che vengono installati nel server. Vi sono poi due altri tipi di connessioni virtuali che servono a far comunicare due partizioni fra di loro quando queste si trovano sullo stesso server:

- **Internal Coupling Links:** connessioni per il Parallel SYSPLEX all'interno dello stesso server con protocollo proprietario

- **HiperSocket**: connessioni virtuali interne al sistema per collegare partizioni logiche e macchine virtuali (cioè Guest zVM) fra di loro. Supportano il TCP/IP al layer 3 (IP) o SNA e TCP/IP al layer 2 (MAC). Velocità superiore a 300 MB/sec.

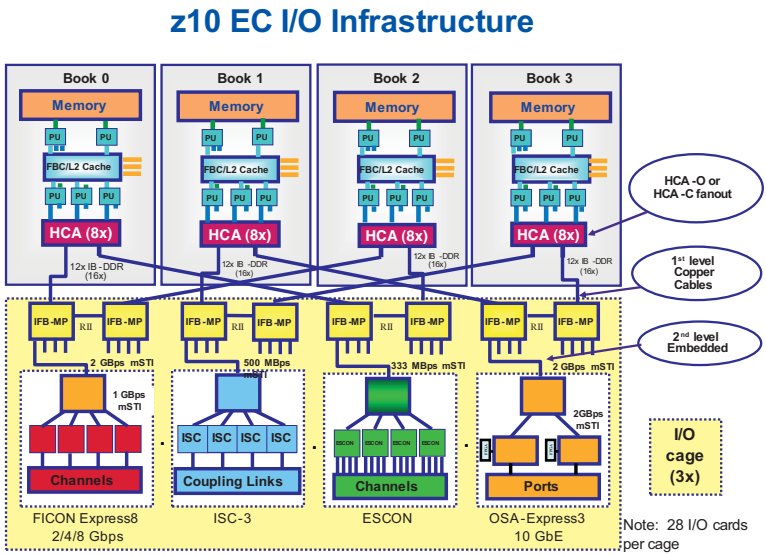


Figura 14 Le connessioni book – I/O cage

Poiché la memoria si trova nei book, come si collegano le interfacce che abbiamo descritto ai book? Questo compito è affidato essenzialmente agli Host Communication Adapter (HCA2-C) in rame che si trovano in ogni book. La figura 14 illustra le connessioni interne fra i book, HCA2-C e I/O cage.Gli adapter HCA2-C hanno 2 link Infiniband. Ciascun link ha la portata di 6 Gbyte/sec. Gli HCA2-C sono connessi a tutte le I/O cage presenti nel server. Si noti che, mentre il contenuto di un book è prefissato, il contenuto delle I/O cage varia a seconda della configurazione di I/O (Quanti e quali adapter) installata nel server. Si possono avere fino a 3 I/O Cage; ogni I/O Cage può contenere fino a 28 schede di I/O (vengono chiamate anche Blade a causa della loro forma molto sottile). Ogni scheda è monotipo: non si possono avere sulla stessa scheda interfacce diverse (ad esempio, non si può avere ESCON + FICON sulla stessa scheda).

La portata di 6 Gbyte/Sec di ogni link infiniband viene distribuita fra tutte le schede presenti nella configurazione. Il numero di HCA2-C presenti nel server dipende da:

- Numero dei book presenti
- Numero di schede di I/O presenti nella configurazione.

2.0.7 Support Element e HW Management Console

Un server oggi è così complesso nella sua struttura HW che ha bisogno di altri computer per essere gestito. Questo ruolo viene svolto sui mainframe da un PC (portatile) che viene chiamato Support Element (SE) - vedi figura 15. Le sue funzioni si raggruppano in due macro aree:

- Controllo dei parametri ambientali (Temperatura, calore generato e smaltito, Potenza elettrica assorbita)

- Gestione della configurazione del server. Abbiamo già visto che sul SE viene scritto l'IOCDS. Vi sono tante altre informazioni, quali ad esempio la configurazione delle partizioni logiche, le opzioni di partenza dei vari sistemi operativi che si vogliono attivare in questa o quella parizione logica e così via. Inoltre raccoglie il log di tutti gli errori HW che si verificano sul server per inviarli, se necessario, ai centri IBM per risolvere i problemi. Tutte queste informazioni sono memorizzate sul suo disco fisso.

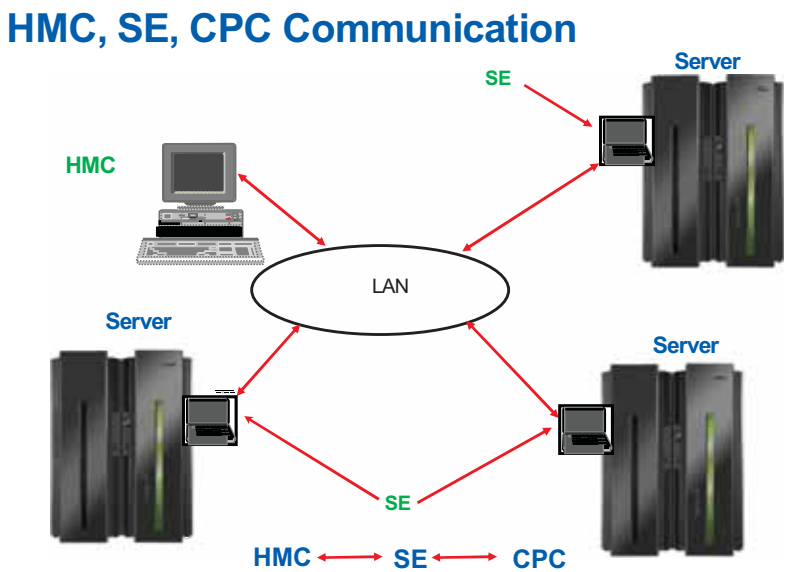


Figura 15 2HMC, SE, Server

Il Support Element è necessario per ogni attivazione/disattivazione di elementi del server; in partivcolare è necessario per accendere il server (naturalmente SE ha un'alimentazione separata), per attivare o disattivare partizioni logiche, per far partire i sistemi operativi, e altre attività ancora.

Quindi il SE è indispensabile. Per questo motivo ve ne sono due in ogni server come si vede nella figura 16.

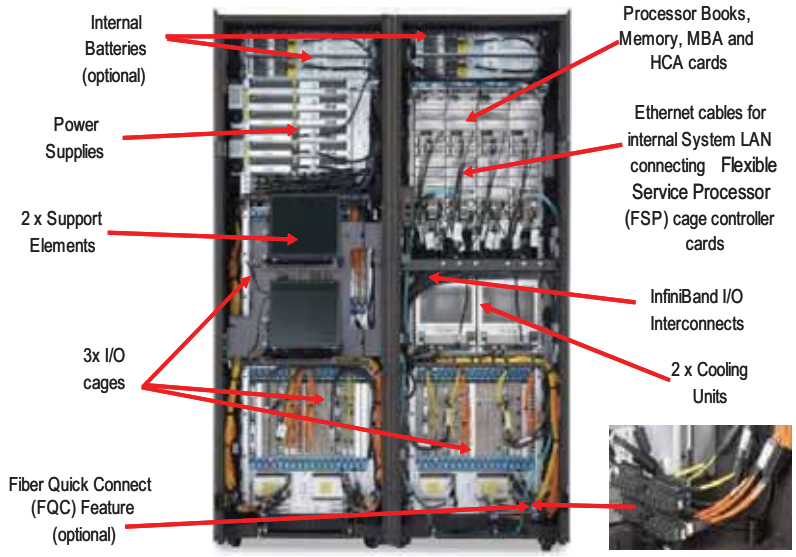


Figura 16 Interno del sistema z10

I SE si trovano all'interno del server. Inoltre un SE può governare solo il server in cui è collocato, e questo è normalmente collocato in posizioni protette e di difficile accesso. Vi sono parecchie installazioni in cui sono presenti più di un server. E' utile avere un modo di controllare tutti i mainframe da un unico punto: questa funzione viene assolta dalla HW Management Console. Sempre in figura 15 vediamo una situazione tipica: abbiamo 3 Server, ognuno con il suo SE. Questi sono collegati in LAN con una HW Management Console, dalla quale è possibile controllare i tre server. Attenzione: l'HMC non sostituisce i SE; infatti se un SE non fosse raggiungibile dall'HMC, quel server potrebbe essere gestito solo agendo sul suo SE. Ogni comando impartito dall'HMC verso un certo server viene trasmesso al SE corrispondente che agisce sul server.

Come esempio dello sviluppo del Mainframe illustriamo due evoluzioni molto significative che interessano le connessioni Fiber Channel:

- zHPF - *High Performance FICON per System z.*
- *Fiber Channel Protocol for SCSI*

2.0.8 High Performance Ficon

E' stato introdotto nell'Ottobre 2008 sui modelli maniframe z10 ed è un'estensione della z/Architecture chiamata Fiber Channel Extension. Lo scopo è aumentare sia il numero di Operazioni di I/O che il numero di Gbyte trasferiti nell'unità di tempo. Lo

scopo essenziale è quello di migliorare le prestazioni di I/O cambiando la modalità con cui vengono scritti e eseguiti i programmi di canale, riducendo in questo modo l'overhead sul canale, sulle porte delle Control Unit (CU) e sul link.

Per capire come zHPF migliora le prestazioni possiamo fare riferimento al programma canale illustrato nel capitolo sull'Architettura. Il funzionamento standard dell'architettura che abbiamo illustrato ha molti pregi ma non quello della rapidità. Come avveniva la comunicazione Canale Control Unit con il protocollo FICON, prima dell'introduzione di zHPF?

Un channel program consiste in una serie di *Channel Command Words* -CCWs- concatenate, nelle quali viene indicato il tipo di operazione, il numero di bytes da trasferire etc. Ognuna di queste CCW rappresenta una FICON *channel Information Unit* –IU- che deve essere processata singolarmente.

Al contrario, per facilitare e velocizzare il processo di I/O, l'architettura zHPF definisce un singolo blocco di comandi - *Transport Control Word*, TCW- che sostituisce una serie di CCWs, come mostrato in figura 17.
Si è in grado in questo modo di creare un *multiple channel commands Transport* da spedire come un'unica entità, anzichè spedire comandi singoli come avviene nel protocollo FICON, fornendo un protocollo molto più semplificato.
In figura 17, nella parte sinistra, è mostrato un esempio di un programma di canale che legge blocchi da 4k, dove tre FICON IUs sono spedite dal canale alla CU e altre tre IUs dalla CU al canale.

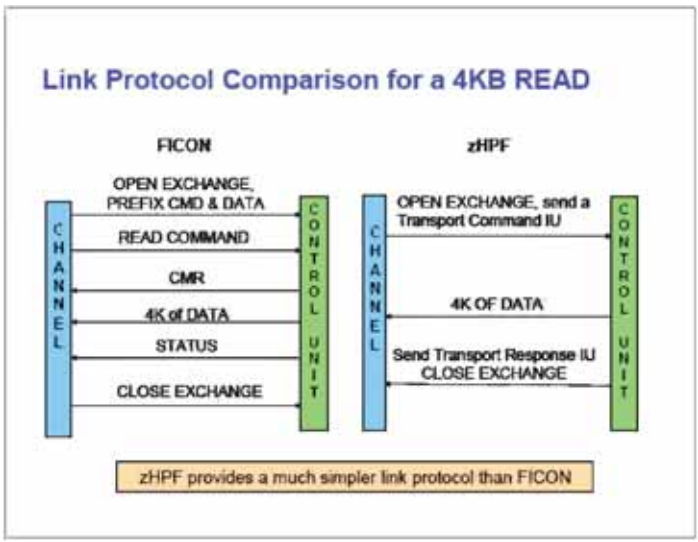


Figura 17 Confronto tra protocollo FICON e zHPF

Nella parte destra della stessa figura, si può notare come il protocollo zHPF riduca della metà il numero di IU spedite, utlizzando solo una IU dal canale alla CU e due

dalla CU al canale; zHPF riduce dunque una stringa di CCW in una singola nuova *Control Word*.
zHPF può essere attivato semplicemente agendo su parametri all'interno del sistema operativo. E' una funzione da attivare selettivamente poichè esige un supporto complementare sullo storage subsystem.
I canali FICON Express8 (FEx8), FICON Express4 (FEx4) e FICON Express2 (FEx2) supportano questo protocollo; in particolare, FICON Express8, sfruttando il protocollo zHPF, arriva ad ottenere 730 MegaBytes per secondo (MBps) di *throughput* per trasferimento di dati *full-duplex* anzichè 510 MBps, che è il throughput massimo che si ottiene utilizzando il FICON tradizionale.

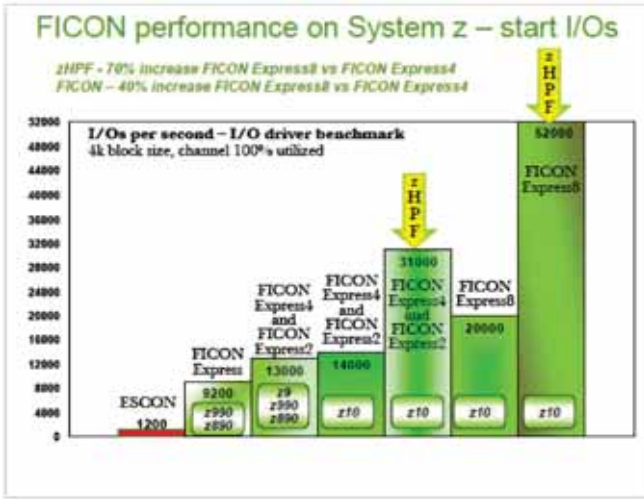


Figura 18 Confronto tra le performance dei canali FICON, con e senza zHPF

Nella figura 18 vediamo invece come aumenta il numero di Operazioni di I/O che possono essere eseguite. Il canale FICON Express8 su server System z10 con protocollo zHPF, arriva fino a 52,000 IO/sec (blocchi di 4k). Comparato con FICON Express4, questo rappresenta un miglioramento di circa il 70% quando è attivo il protocollo zHPF e del 40% quando viene utilizzato il protocollo FICON tradizionale.

2.0.9 Fibre Channel Protocol per SCSI

L'architettura System z *Fibre Channel Protocol* – FCP è conforme allo standard specificato dall' INCITS - *InterNational Committee of Information Technology Standards*.

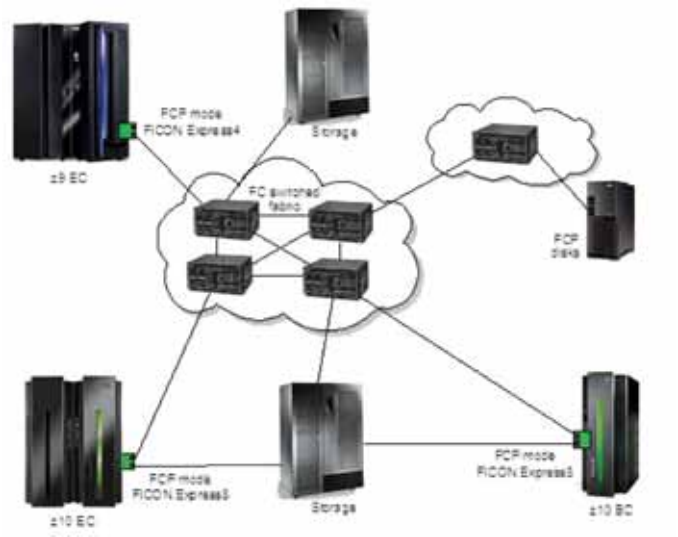


Figura 19 Un mainframe inserito in una SAN usando il protocollo FCP

FCP supporta l'accesso alle periferiche *Small Computer System Interface* -SCSI- in ambiente z/VM, z/VSE, e Linux su System z aumentando in questo modo la scelta della soluzione storage come mostrato in figura 19. Ogni canale FICON Express8, FICON Express4, FICON Express2, e FICON Express può essere definito in modalità FCP: il FICON channel in questo caso viene configurato con CHPID type FCP. Tutto questo è in alternativa al protocollo FICON, che fornisce il supporto per l'accesso ai device periferici di tipo *Extended Count Key Data* – ECKD (cioè organizzati secondo l'architettura dati propria del mainframe).

FCP utilizza l'architettura *Queued Direct Input/Output* (QDIO), in maniera del tutto simile a quella implementata nell'*Open Systems Adapter* (OSA), permettendo un trasferimento diretto *memory-to-memory* tra host e canale. L'unico requisito è che il canale definito FCP deve essere collegato ad una Storage Area Network (SAN). Non sono supportate configurazioni Point to Point non switched. I sistemi operativi che supportano questa connessione sono z/VM e zLinux.

Poter far funzionare un canale FC in modalità FCP è interessante per due motivi:

- Quando si vogliono consolidare dei mondi Linux che girano su piattaforma Intel sul mainframe una delle attività da fare (se non ci fosse questa possibilità) è di trasformare i dati su storage dal mondo Open al mondo Mainframe. Questa è una attività delicatissima e suscettibile di errori molto frequenti. Invece se il mondo Linux su Mainframe usa dei canali FCP non c'è bisogno di nessuna trasformazione; basta avere alcune porte libere sulla SAN per collegare gli Adapter FCP usati da Linux.
- Si possono utilizzare dispositivi storage non creati per il mondo Mainframe ampliando molto la scelta HW per far risiedere i dati.

2.1 Hardware di IBM zEnterprise

IBM il 22 luglio 2010 ha annunciato la nuova famiglia di elaboratori IBM zEnterprise System. Questo sistema è la prima realizzazione di Elaboratore Ibrido poichè include, sotto la gestione di un componente chiamato IBM zEnterprise Unified Resource Manager, processori di piattaforme diverse (z, p ed Intel). L'importanza di questo annuncio per la crescita di applicazioni complesse e la gestione degli ambienti informatici è oggetto di trattazione nell'Appendice A di questo volume.
In questo capitolo si vogliono invece fornire alcuni cenni alle innovazioni specifiche per la componente tecnologica della piattaforma z (IBM zEnterprise 196) facendo riferimento alla precedente famiglia di elaboratori IBM System z10 Enterprise Class.

Il chip del nuovo elaboratore (come per lo z10) è un 4 core con ciascuna Processor Unit circondata da vari livelli di cache interna dedicati al singolo processore o condiviso tra i suoi vicini (figura 20). Con i suoi 5,2 GHz di potenza è il processore più potente utilizzato in elaboratori commerciali.

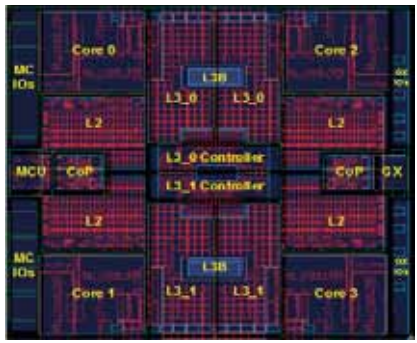


Figura 20 Chip dell'elaboratore IBM zEnterprise 196 (z196)

Il numero dei livelli di cache e la loro dimensione è variata notevolmente rispetto al predecessore z10 EC.

Nella figura 21 i due disegni interni del chip sono messi a confronto: la cache di level4 ha il compito di condividere i dati e le istruzioni tra i diversi chip del z196 Multiple Chip Module (MCM) e tra i gli MCM dei diversi book. É posizionata sui 2 chip di tipo Storage Control (SC) presenti su ogni MCM.

La cache di livello 3 ha invece il compito di condividere informazioni tra i core dello stesso chip. La figura 21 evidenzia la differenza tra le due soluzioni e le dimensioni di memoria utilizzabili negli elaboratori z10 e z196.

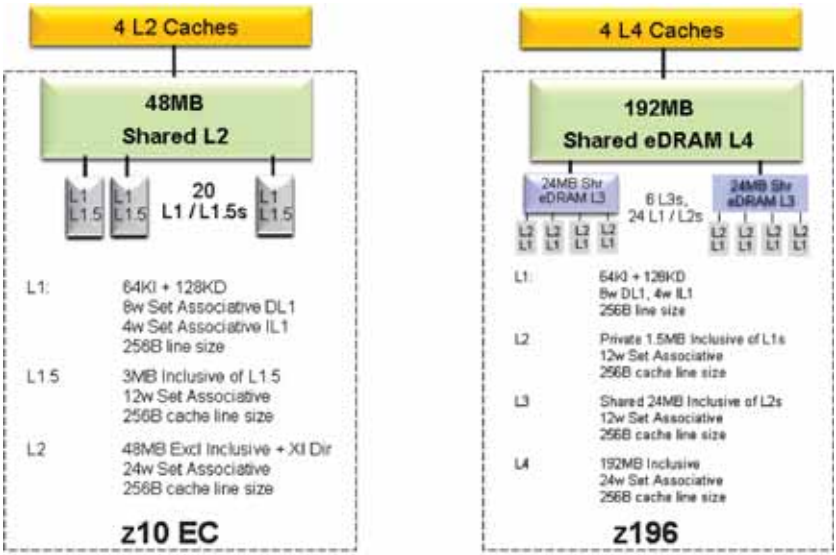


Figura 21 Confronto tra le cache dei processori z10 e z196

Il disegno logico di questa componente fondamentale è stato quindi rivisto per consentire prestazioni più elevate per tutti i tipi di workload presenti sugli elaboratori della piattaforma z. Nella figura 22 sono rappresentati i due schemi utilizzati nei due elaboratori.

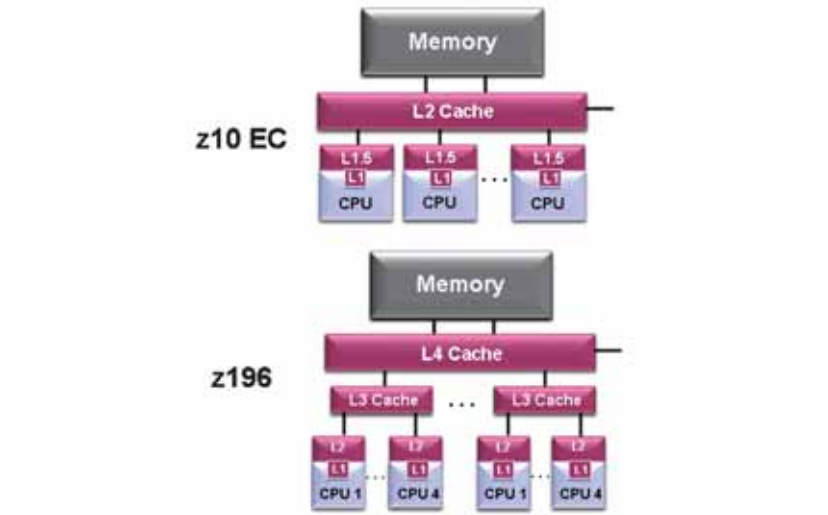


Figura 22 Disegno logico di utilizzo della cache interna

Negli MCM dello zEnterprise 196 i chip di tipo PU sono 6 (contro i 5 dello z10) e possono contenere ciascuno 20 o 24 processori attivi; i chip di tipo Storage Control (SC) sono ancora 2 ed all'interno di questi è disponibile la funzione di Master Time-of-Day (TOD) che ha alcuni elementi anche in ogni chip PU. I processori SAP (Service Access Processor) e gli Spare (Spare Processor) si affiancano a quelli visibili ai sistemi operativi e personalizzabili nelle diverse configurazioni. Nella figura 23 è raffigurato un MCM dello z196 con evidenziati i differenti chip.

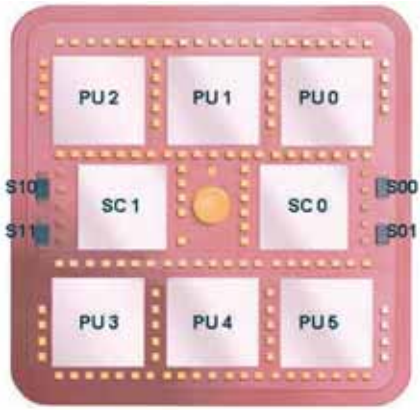


Figura 23 MCM (Multiple Chip Module) di zEnterprise 196

Le differenze tra lo MCM dello z196 ed quello del suo predecessore sono riassunte nella figura 24 che ne evidenzia anche le dimensioni ed il consumo energetico.

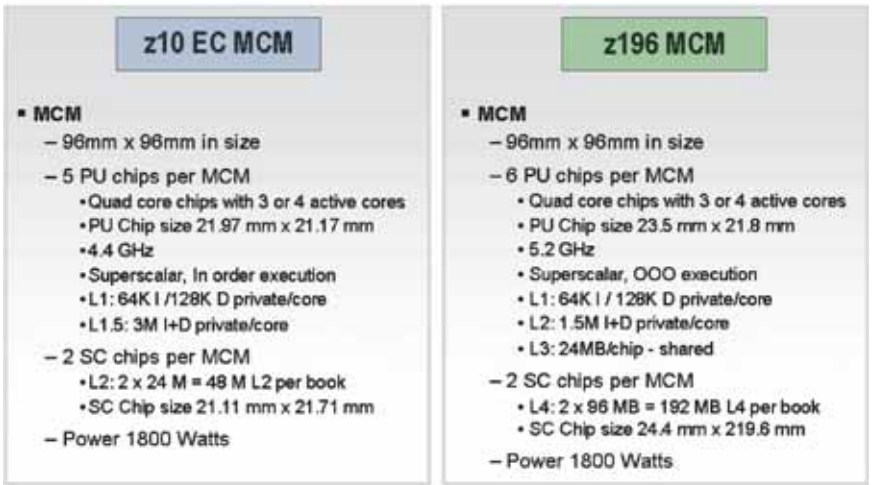


Figura 24 Confronto tra MCM di zEnterprise 196 e z10

In base al numero di book installati (e quindi ai processori disponibili ai sistemi operatvi) si possono avere modelli diversi. I modelli sono 5 di cui 4 composti da MCM con "20 – core" (due chip forniscono 4 core e 4 solo 3 core):

Modello	PU applicativi	SAP	Spare	Totale
M15	15	3	2	20
M32	32	6	2	40
M49	49	9	2	60
M66	66	12	2	80

Il modello M80 usa quattro MCM da "24" i cui processori vengono ripartiti in questo modo:

Modello	PU applicativi	SAP	Spare	Totale
M80	80	14	2	96

Una ulteriore innovazione rispetto alle famiglie precedenti è legata alla memoria centrale disponibile le cui dimensioni sono raddoppiate (da 1,5TB in z10 ai 3TB in z196) nella massima configurazione.

La memoria è disponibile in "Dual in-line Memory Module" (DIMM) ed ogni book può avere fino a 30 di questi componenti connessi al MCM tramite 3 Memory Control Unit (MCU). Ognuna di queste MCU raggiunge le DIMM con 5 canali: il quinto serve ad implementare la Redundant Array of Independent Memory (RAIM). Questa tecnologia migliora ulteriormente la capacità di individuare e correggere gli errori in questa risorsa.

Nella figura 25 è rappresentata questa nuova modalità di connessione tra Memoria Centrale e processori.

Maggiori dettagli sulle caratteristiche tecniche dello zEnterprise 196 sono disponibili nei siti IBM, quali ad esempio:

http://www-03.ibm.com/systems/z/news/announcement/20100722_resources.html

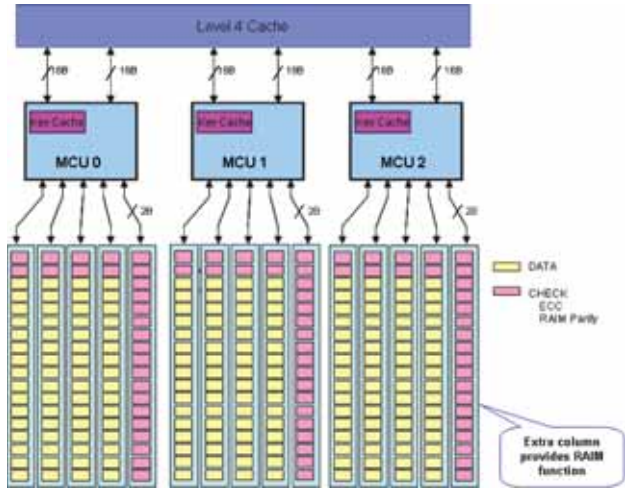


Figura 25 La memoria RAIM dello zEnterprise 196

2.2 La Virtualizzazione

La virtualizzazione delle risorse informatiche rappresenta una caratteristica essenziale dei sistemi informatici. Infatti è oggi imperativo massimizzare l'utilizzo dei Sistemi; inoltre i sistemi informatici debbono potersi adattare alle variazioni anche rapide del Carico Applicativo, sia dal punto di vista della quantità che del tipo di carico di lavoro. Quest'ultima caratteristica è detta comunemente "resilienza" (resiliency): grazie ad essa si può garantire un livello di servizio accettabile nonostante l'imprevedibilità del carico dovuta alla estrema estensione delle applicazioni accessibili in rete da un grande numero di "potenziali" utenti, non noto a priori.

2.2.0 Concetti Generali

Vi sono varie definizioni della virtualizzazione; ne citiamo due molto frequentemente usate:

- La capacità di creare, usando una componente hardware o software, immagini di risorse informatiche virtuali sulla base delle risorse fisiche realmente disponibili; ogni immagine rappresenta un sottoinsieme logico delle risorse reali.
- "Virtualizzazione è il processo di presentazione delle risorse di un sistema di calcolo in maniera tale che gli utenti e le applicazioni possono facilmente averne un beneficio, invece di rappresentarle in base alla loro implementazione, dislocazione geografica o composizione fisica. In altre parole, viene fornita una visione logica invece che fisica dei dati, potenza di calcolo, capacità di immagazzinamento ed altre risorse".

Nelle figure 1 e 2 è rappresentata schematicamente la relazione fra risorse fisiche, tecnologia di vitualizzazione e risorse virtuali.

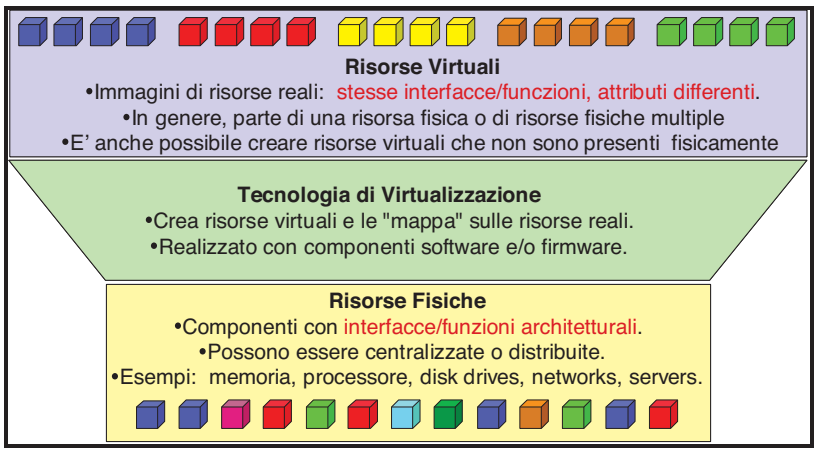


Figura 1 Virtualizzazione

2.2.1 Caratteristiche della Virtualizzazione

- Ogni macchina virtuale, o partizione logica, si presenta nei confronti delle altre come se fosse fisicamente separata.
- Indipendenza: ogni evento che si verifica su una macchina virtuale, o partizione logica, non deve ripercuotersi né interferire sulle attività delle altre.
- V=R: all'interno di un sistema a più macchine virtuali, o partizioni logiche, deve essere sempre permesso riservare risorse fisiche ad una o più di esse. Tali risorse vengono identificate come Risorse Dedicate.
- Unico Punto di Gestione: un sistema con più macchine virtuali, o partizioni logiche, ha un unico punto di accesso in grado di gestire e controllare le macchine virtuali e la loro configurazione.

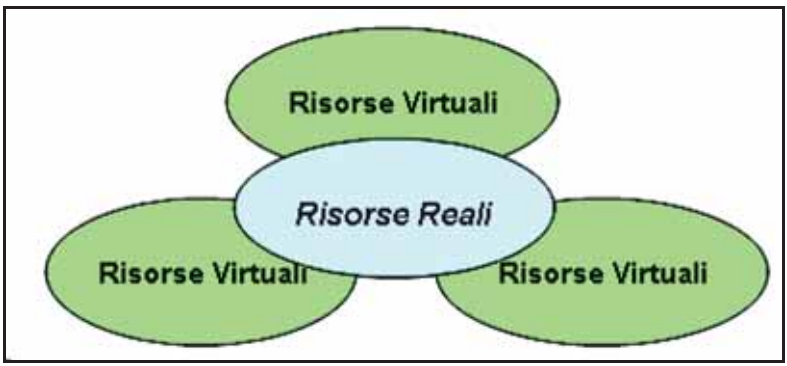


Figura 2 Risorse reali e virtuali

2.2.2 Le ragioni della Virtualizzazione

I motivi che hanno fatto sviluppare la tecnologia della virtualizzazione sono essenzialmente legati al tentativo di ridurre le dimensioni delle infrastrutture informatiche ed ottimizzarle, poiché diventa sempre più necessario ridurre al minimo i costi di gestione delle stesse.

Uno dei benefici della virtualizzazione è la possibilità di creare nuovi sistemi senza dover acquistare nuovi dispositivi o nuovi server. Ad esempio, in questa maniera sono soddisfatte le esigenze di avere nuovi sistemi che ospiteranno ambienti per un periodo limitato di tempo, come sistemi di sviluppo legati ad un determinato progetto; alla sua conclusione sarà possibile "distruggere" le macchine virtuali senza dover "distruggere" di conseguenza alcun dispositivo hardware. Dal punto di vista economico, questo è

sicuramente un grosso beneficio. Riassumendo, abbiamo due caratteristiche fondamentali per la virtualizzazione:

- Ottimizzazione - per definizione, la somma delle risorse virtuali definite può essere maggiore delle risorse effettivamente presenti e di conseguenza è possibile utilizzare risorse virtuali in eccesso. Questo fenomeno viene definito "over committment" oppure "over booking".

- Separazione - con la Virtualizzazione si possono suddividere le risorse reali in ambienti logicamente separati gestiti in maniera indipendente.

L'esempio che segue fa vedere come la virtualizzazione possa essere un efficiente metodo per ottimizzare ed ottenere un beneficio sia tecnico che economico.

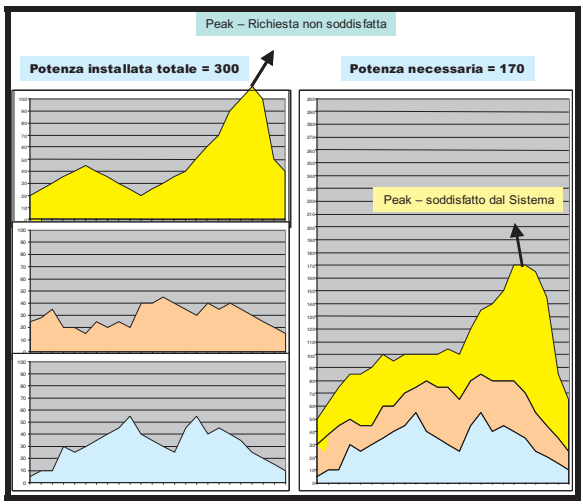


Figura 3 Condivisione = Risparmio

Nella parte sinistra della Figura 3 sono riportati i grafici che mostrano l'attività giornaliera di 3 sistemi diversi che ospitano altrettanti workload. Prescindendo dall'unità di misura, assumiamo che ogni sistema abbia una capacità installata pari a 100, per cui la potenza totale installata è pari a 300. Come si può notare, il primo sistema (giallo) non riesce ad onorare il picco di carico.
Sulla destra, è stato rappresentato il grafico, stessa scala, di un sistema virtualizzato che ospita tutti e 3 i workloads precedenti. Come si può notare, la potenza necessaria richiesta è notevolmente inferiore, 170/300, e soprattutto il sistema dimostra una notevole flessibilità nell'assorbire i picchi di carico.

2.2.3 Le risorse virtualizzabili

Le risorse virtualizzabili sono le seguenti:

- **CPU** - Più macchine virtuali possono condividere lo stesso processore. A seconda delle implementazioni la granularità della condivisione può essere più o meno profonda; nelle implementazioni software z/VM ed ESX questa granularità può essere teoricamente infinita, praticamente limitata dalla potenza del processore. Tutte le implementazioni IBM ed ESX consentono dinamicità di assegnazione della risorsa CPU anche durante il funzionamento, in base alle richieste dei Sistemi Ospiti;

- **Memoria** - In tutte le implementazioni Hardware IBM e non IBM la memoria è fisicamente assegnata ad una Partizione Logica o Fisica e può essere riconfigurata sia pure a Sistemi attivi, ma solo con un intervento esterno; essa pertanto non viene virtualizzata. Nelle implementazioni Software z/VM, ESX, vPars, la memoria è virtualizzata in quanto risorsa gestita dal Virtualizzatore che possiede suoi meccanismi di Memoria Virtuale, di paginazione e swapping. Se necessario, con z/VM è possibile assegnare della memoria reale direttamente ad un Sistema Ospite (V=R): in questo caso la memoria assegnata è gestita direttamente dal Sistema Ospite e viene esclusa dai meccanismi di paginazione del Virtualizzatore.

- **Dischi** - Nelle implementazioni Software sono sempre viVirtualizzati mentre nelle implementazioni Hardware può essere virtualizzata la connessione Server - Storage, con modalità diverse a seconda della piattaforma.

- **Altri dispositivi di I/O** - E' un argomento di cui il mercato si occupa poco anche se molto importante. Immaginiamo di avere una scheda di rete a 6GBit che sia dedicata ad una partizione logica che la utilizza solo al 50%. La partizione adiacente, cui è stata assegnata una scheda di rete da 2Gbit, non riesce a smaltire tutto il traffico con gli utenti. Se si condivide la scheda di rete a 6Gbit fra le due partizioni si ha banda passante sufficiente a entrambe le partizioni e si risparmia la scheda da 2Gbit. Sia la virtualizzazione HW che la virtualizzazione SW sono in grado di realizzare questo con modalità diverse nella gestione degli IP address e dei MAC Address.

2.3 Virtualizzazione HW e SW

Abbiamo introdotto nel paragrafo precedente le nozioni di Virtualizzazione HW e Virtualizzazione SW. Entrambe hanno l'obiettivo di realizzare la virtualizzazione, entrambe creano delle immagini virtuali delle risorse fisiche.
La differenza fra i due approcci sta nell'implementazione e nel tipo di funzioni che vengono offerte. Le principali differenze fra i due modi sono:

- L'implementazione HW non si basa su alcun Sistema operativo; essa è implementata nell'HW con opportune aggiunte di Firmware. Al contrario

l'implementazione SW è incapsulata in sistemi operativi (z/VM o VmWare ESX ne sono un esempio).

- La virtualizzazione SW offre più funzioni alle immagini virtuali della virtualizzazione HW.
- Il costo in termini di potenza di calcolo spesa per gestire il complesso delle immagini virtuali è maggiore con la virtualizzazione SW che con la virtualizzazione HW.
- Per consuetudine un insieme di risorse virtuali realizzato mediante HW si chiama **Partizione Logica**. Si chiama **Macchina Virtuale** l'insieme di risorse virtuali realizzato mediante SW.

In questo capitolo descriveremo la virtualizzazione HW come è realizzata sui Mainframe z Series. Lasceremo la descrizione della virtualizzazione SW ad un capitolo apposito in cui si parlerà del sistema operativo z/VM.

2.3.0 Le diverse opzioni disponibili

La virtualizzazione HW può realizzarsi in due modi come illustrato in figura 4; in questo caso la tecnologia di virtualizzazione prende il nome di Partizionamento Logico. Il partizionamento logico può essere realizzato con il firmware, o hypervisor, Type1, o con firmware e sistema operativo, Type2.

Il Type1 è diventata l'opzione dominante nel mercato dei server per l'alta efficienza ed affidabilità.

La soluzione Type2 si rivolge essenzialmente al mercato delle soluzioni workstation e clients, dove è richiesta un'alta integrazione con il sistema operativo.

Altro elemento di differenziazione all'interno dei virtualizzatori basati su Firmware o su Hypervisors (Type1) è rappresentato dalla presenza o meno di particolari specifiche istruzioni (o set di istruzioni) legate alla virtualizzazione all'interno della architettura della CPU.

Ad esempio le CPU basate sulla z/Architecture contengono una istruzione denominata SIE (Start Interpretive Execution) in grado di facilitare la creazioni di Partizioni Logiche e Sistemi Virtuali basandosi anche sull'uso di particolari dispositivi Hardware presenti nelle CPU stesse (Special Register). Tale tecnica è in via di estensione anche ad altre architetture.

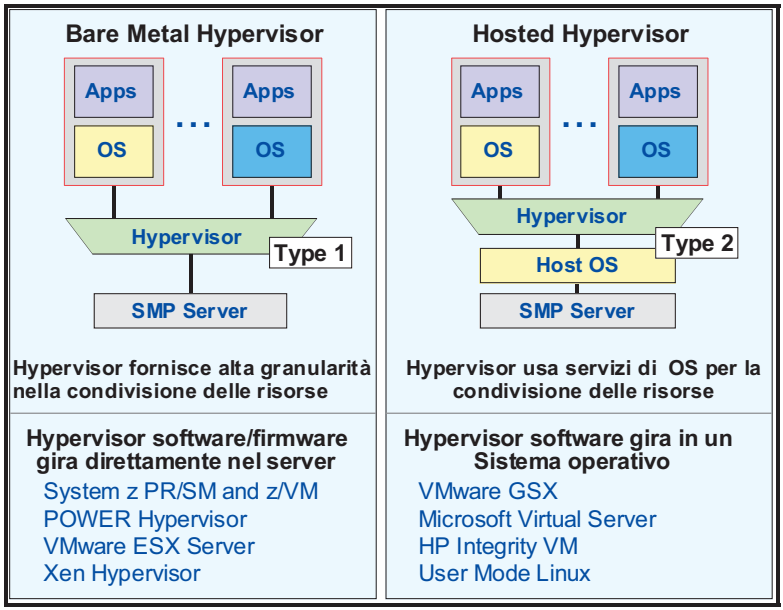


Figura 4 Virtualizzazione HW e Partizionamento logico

Nel caso di presenza di dispositivi Hardware specifici e delle relative istruzioni il processo di virtualizzazione risulterà più efficiente (Hardware Virtualization) rispetto al caso contrario (Paravirtualization).

2.3.1 Il Partition Resource & System Management (PR/SM)

PR/SM è il nome assegnato al partizionatore logico creato per i sistemi z. Il PR/SM divide un server in un certo numero di Partizioni Logiche (Logical Partition - LPAR). Ognuna di queste LPAR è totalmente indipendente ed isolata dalle altre LPAR ed in essa è attiva una unica istanza di Sistema Operativo.
Il PR/SM è un firmware microcodificato e non modificabile dall'utente e, in quest'accezione, può essere considerato un dispositivo Hardware. La funzione principale è di considerare le risorse fisiche (CP, Memoria, Canali) come un unico shared-pool e di permettere a tutte le Partizioni Logiche (LPAR) definite di considerare le risorse logiche ad esse assegnate come reali e di uso esclusivo. La figura 5 mostra un esempio di macchina z configurata con partizioni logiche e macchine virtuali (sotto z/VM).

La definizione migliore per il PR/SM è la seguente: "un dispositivo standard, hardware e LIC[1], disponibile sugli elaboratori IBM System z che permette ad una macchina fisica l'esecuzione contemporanea di più sistemi operativi".

[1] Licensed Internal Code

Principali caratteristiche del PR/SM:

- Condivisione completa delle risorse
- Supporto simultaneo di sistemi operativi diversi
- Time of Day (TOD) diversi per le diverse Partizioni Logiche
- Allocazione dinamica delle risorse
- Risorse dedicate - se necessario
- Limitazione delle risorse disponibili a certe LPAR (Capping)
- Carichi di lavoro misti in ogni LPAR
- Ri-definizione delle risorse assegnate in base alle esigenze
- Granularità delle risorse

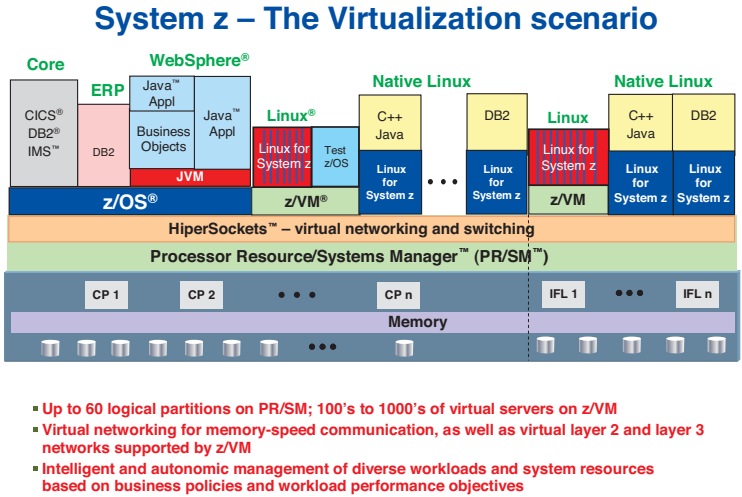


Figura 5 PR/SM Highlights

La figura 6 riassume il numero massimo di partizioni logiche, per ogni famiglia di elaboratori IBM, che il PR/SM permette di ospitare.

Uno degli aspetti più qualificanti del PR/SM è la granularità offerta in termini di suddivisione della risorsa processore. Infatti è possibile assegnare potenze piccole a piacere a ogni partizione logica. Di fatto però ci si scontra con i limiti fisici degli algoritmi di controllo del PR/SM per cui IBM dichiara che la distribuzione della potenza secondo i pesi avrà un certo scarto rispetto alla definizione, scarto che sarà tanto maggiore quanto minore è il peso assegnato alla partizione.

Famiglia	Max Lpar
z900 , z800	15
z990 , z890	30
System z9 + z10	60

Figura 6 Numero Max LPAR

2.3.2 PR/SM - Caratteristiche delle partizioni logiche

Una partizione logica si presenta come un sistema isolato costituito da processore, memoria ed I/O. Ad ogni partizione è assegnato un nome, un numero di processori logici, peso, memoria centrale (ed espansa), canali, TOD clock e specifiche per l'accesso alle interfacce di I/O (canali). Per ogni tipo di risorsa è possibile assegnare diversi parametri e definizioni.

Prendiamo, ad esempio, la risorsa processore. I processori logici possono essere:

- Dedicati
  - in rapporto 1/1 con un processore fisico
  - appartengono in maniera esclusiva a quella partizione
- Shared
  - in rapporto 1/n con un processore fisico
  - la condivisione con le altre partizioni è in funzione del "peso" assegnato alla partizione logica
  - è possibile superare il peso assegnato solo se le altre partizioni non richiedono l'uso del processore
  - il Capping Hardware non permette di superare il "peso" assegnato
  - il Capping Software permette di superare la capacità assegnata per brevi periodi.

Analizziamo in dettaglio alcuni aspetti, quali il peso ed il capping. Ad ogni partizione è possibile assegnare un valore di peso da 1 a 999, e tale valore viene preso in considerazione dalla componente di dispatching del PR/SM in caso di contesa della risorsa processore. Vediamo un esempio in figura 7.

3 partizioni
6 processori fisici
8 processori logici

Partizioni	Proc logici	Peso	LP/PP	Peso Relativo	%
LparA	2	300	2/6	300/1300	23,1%
LparB	1	100	1/6	100/1300	7,7%
LparC	5	900	5/6	900/1300	69,2%
Totale	8	1300			100%

Figura 7 Esempio di definizioni

In questo caso, su un sistema con 6 processori fisici ed 8 processori logici definiti, la partizione **LparC** avrà garantito l'uso del 69,2% (900/1300) della potenza totale disponibile. Si noti che, sempre per LparC, il rapporto processori logici/processori fisici permetterebbe di arrivare all'83,33% della potenza della macchina. Poiché il suo peso è il 69,2%, LparC potrà arrivare all'83,33% solo se le altre due LPAR non useranno tutta la potenza loro assegnata dai loro pesi.

Come detto in precedenza, la potenza non utilizzata o non richiesta da una partizione viene lasciata libera per le altre le quali possono usare più potenza superando lo "share" loro assegnato. Il Capping, hardware o software, serve a contenere il consumo di potenza di una partizione nel rispetto del peso ad essa assegnata.

2.3.3 PR/SM - Gestione I/O - Multi Image Facility (MIF)

Le prime versioni del PR/SM consentivano la condivisione della risorsa processore ma non prevedevano ancora la piena condivisione delle interfacce di I/O, i canali. Era necessario quindi, per permettere l'accesso ad un dispositivo esterno, device, alle N partizioni logiche, prevedere N accessi diversi. Tutto ciò portava ad un sovradimensionamento e ad una ridondanza che non consentiva un efficiente gestione dei sistemi informatici.

Nel **1992** fu introdotto il Multiple Image Facility (MIF) per risolvere il problema dei canali replicati in un ambiente partizionato, consentendo la condivisione dei canali tra le partizioni logiche.

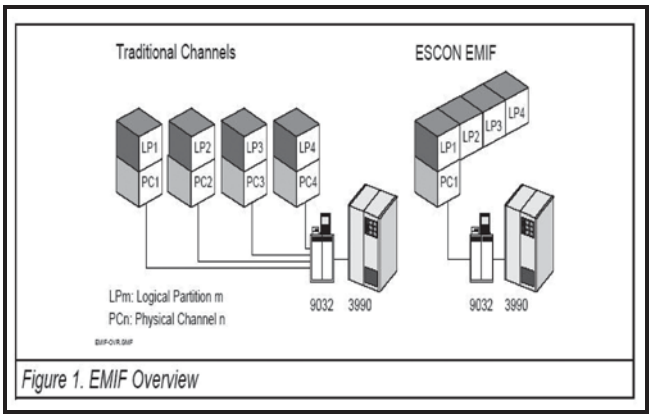


Figura 8 MIF

Il benefici del MIF sono riduzione del numero dei canali necessari, maggiore efficienza nell'utilizzo dei canali e facilità nella gestione delle configurazioni. La Multi Image Facility è realizzabile grazie a due condizioni:

- alle caratteristiche architetturali della gestione delle operazioni di I/O sui sistemi mainframe (aspetto architetturale).
- Introduzione di interfacce di I/O basate sull'uso di frame che possono contenere l'indicazione della partizione che richiede l'operazione di I/O.

Per sfruttare questa caratteristica è necessario assegnare a ogni partizione logica definita un identficatore: Multiple Image Facility Identifier (MIF-Id).

2.3.4 Risorse di I/O virtuali (HiperSocket)

Il PR/SM consente la virtualizzazione delle connessioni LAN tra le partizioni logiche all'interno dello stesso server. Questa modalità viene chiamata HiperSockets. La velocità di questo tipo di connessione, regolato dal protocollo TCP/IP, è estremamente elevata in quanto lo scambio di messaggi avviene direttamente in memoria, per cui la velocità è in relazione al tempo d'accesso alla memoria stessa. Con questa modalità, a parte la notevole differenza in termini di throughput, vengono eliminate tutte le connessioni fisiche esterne (ad esempio cavi, ecc.), i relativi delay e le possibilità di guasto associate.

2.3.5 La potenza dei Mainframe virtualizzati

Ci semba opportuno, a questo punto, alla fine dei capitoli che hanno descritto i vari aspetti dell'HW Mainframe (Architettura, Implementazione, Virtualizzazione HW) e prima di entrare nella descrizione dei sistemi operativi costruiti per il mainframe parlare di due aspetti delle prestazioni dei mainframe:

- La misura della potenza di calcolo
- L'effetto che il numero dei processori in un server ha sulla potenza di calcolo, cioè dell'Effetto MP (MultiProcessor).

La misura della potenza di calcolo

IBM effettua benchmark molto approfonditi sulle macchine mainframe e pubblica delle tabelle riassuntive che prendono il nome di Large System Performance Reference (LSPR). Queste tabelle si possono trovare tramite Google con una ricerca per la frase "Large System Performance Reference". Questi valori esprimono la valutazione IBM delle capacità relative del processore in un ambiente non limitato da vincoli di sistema (ovvero eliminati tutti i colli di bottiglia, fatta eccezione per la potenza della CPU) per carichi specifici di benchmark definiti e stack software specificato nelle tabelle. La tabella è costruita assegnando alla potenza di calcolo di un determinato modello (definito in testa alla tabella) il valore 1, e esprimendo le potenze di tutti gli altri modelli in rapporto alla potenze del modello base. La tabella comprende tante righe quanti sono i modelli disponibili e tante colonne quanti sono i Benchmark che vengono effettuati. Ad esempio per il sistema z10 abbiamo 64 righe poiché il numero di processori di un sistema z10 varia da 1 a 64. I vari benchmark differiscono per il tipo di applicazione che viene misurato: abbiamo ad esempio carichi transazionali di tipo bancario che usano certi Transaction Monitor e certi Data Base Manager (l'interazione con gli utenti finali è simulata con strumenti SW), carichi Batch (che cioè non simulano interazioni con gli utenti finali), su vari sistemi operativi. Questa varietà di benchmark si rende necessaria poiché la potenza di calcolo che una piattaforma può esprimere dipende pesantemente dal tipo di carico di lavoro che su quella piattaforma viene eseguito. Inoltre i Benchmark sono sempre effettuati con le macchine partizionate logicamente, in quanto questo è il modo attuale di operare di ogni installazione Mainframe.

Nell'effettuare i benchmark si conducono le cose in modo che l'utilizzo della CPU sia attorno al 90%. Questa procedura viene adottata per due motivi:

- 90% è un valore tipico di utilizzo delle piattaforma mainframe
- l'efficienza del sistema di cache (che viene misurata come Hit Rate – quante volte il dato che chiedo alla RAM è stato trovato nella Cache) dipende in grande misura anche dall'utilizzo medio del processore. La conseguenza di questo fenomeno è che se il dato della potenza di calcolo di un modello è stato ottenuto con un utilizzo del 90% estrapolare linearmente al 100% non rappresenta mai un grosso azzardo. Mentre è estremamaente fuorviante moltiplicare per 5 un valore ottenuto avendo l'utilizzo della CPU al 20%.

L'unità di misura della potenza di calcolo è espressa come "numero di transazioni completate/sec/% CPU Busy" cioè è un dato di throughput normalizzato sull'utilizzo della CPU osservato durante la prova (il 90% citato prima è un obiettivo; nella realtà si ottengono valore che oscillano di poco attorno al 90% - di qui la necessità della normalizzazione).

L'effetto MP

Uno dei fenomeni tipici dei sistemi ad architettura SMP[2] è quello denominato "effetto MP", cioè la potenza di ogni processore diminuisce ad ogni aggiunta di un processore alla configurazione. Questa diminuzione, o "effetto MP", è dovuta all'overhead associato alla comunicazione necessaria tra i processori e la relativa gestione a carico e del sistema operativo e del firmware PR/SM. Osserviamo il grafico della figura 9.

Notiamo come questo effetto si sia attenuato nel corso dell'evoluzione tecnologica delle famiglie di elaboratori IBM. L'aggiunta del "processore N", il sesto in questo caso, nella prima generazione di processori IBM CMOS, 9672 G1, forniva solo il 72% della potenza del processore; invece, l'aggiunta del "processore N", il sedicesimo in questo caso, nella penultima generazione di processori IBM CMOS, System z9, riesce a fornire il 78% della potenza del processore. Le ragioni dell'attenuazione sono dovute ai miglioramenti del codice del firmware PR/SM, del sistema operativo z/OS e dell'hardware, necessari per supportare un numero sempre maggiore di processori, partizioni logiche e memoria.

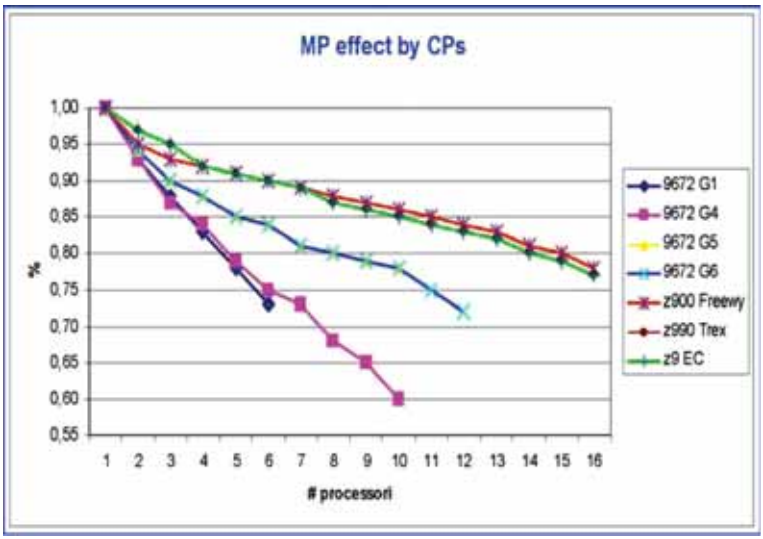


Figura 9 Effetto MP

In Figura 10 abbiamo posto in ascissa il numero di CPU e sulle ordinate i MIPS (una unità di misura della potenza dei mainframe) che la macchina può erogare. Si nota che con l'aumentare delle CPU l'aumento della potenza erogata non è lineare, ma tende a stabilizzarsi (cioè si raggiunge un punto in cui aggiungere motori non fa guadagnare potenza); le cause principali sono:

[2] Simmetric Multi Processor

- Hardware - man mano che si aggiungono processori, le varie CPU si devono sincronizzare tra loro, devono comunicare tra loro, hanno degli elementi comuni (le Cache di livello 2); per garantire l'integrità dell'elaborazione debbono usare dei protocolli che consumano cicli utili.

- Software - il sistema operativo, all'aumentare del numero di CP da gestire, spende sempre più tempo per garantire lintegrità delle elaborazioni. In generale il contributo SW all'MP effect è superiore all'effetto HW.

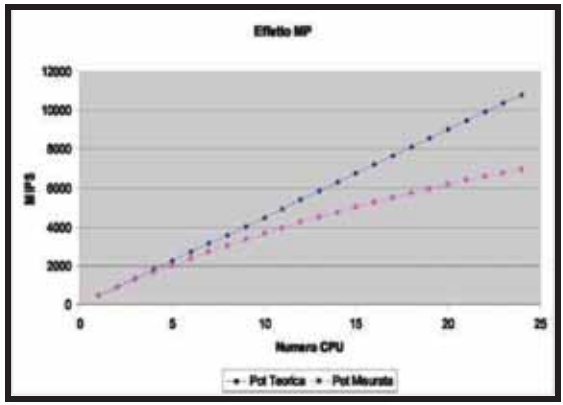


Figura 10 MIPS vs Numero CP

A causa di questo "effetto MP" all'aumentare delle CPU la potenza totale diminuisce rispetto alla potenza teorica. Analizziamo adesso il fenomeno riportato nelle figura 11 e 12. Un elaboratore IBM 2084-320, con 20 processori, eroga una potenza pari a 6000 MIPS (barra azzurra a sinistra), con una potenza per processore pari a 303 MIPS (6060/20).

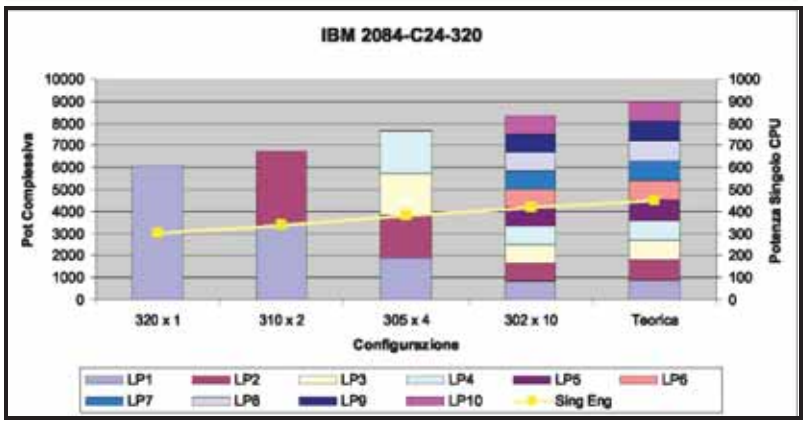


Figura 11 Il partizionamento logico riduce l'effetto MP

Supponiamo di dividere l'elaboratore in due (avremo 2 IBM 2084-310), ognuno di potenza pari a 3370 MIPS , totale 6740 MIPS, potenza per processore di 337 MIPS e così via fino all'ultimo caso dove l'elaboratore è stato diviso in 10 entità da 2 processori ciascuno la cui potenza erogata, 8350 MIPS, è molto vicina alla potenza teorica di 9000 MIPS, 20 x 450 [potenza singolo processore].

	LP1	LP2	LP3	LP4	LP5	LP6	LP7	LP8	LP9	LP10	Sing Eng
320 x 1	6066										303
310 x 2	3370	3370									337
305 x 4	1915	1915	1915	1915							383
302 x 10	835	835	835	835	835	835	835	835	835	835	418
Teorica	900	900	900	900	900	900	900	900	900	900	450

Figura 12 Come le LPAR riducono l'effetto MP

Come si può evincere dal grafico, la potenza ricavabile dalla somma delle singole LPAR è maggiore della potenza complessiva che si potrebbe ricavare dal sistema se esso venisse usato come un unico SMP. Questo avviene poiché la componente SW dell'effetto MP è preponderante rispetto alla componente HW.

La realtà dell'IT è però tale per cui si preferisce comunque crescere con il numero di motori sulla singola partizione logica; infatti per bilanciare un carico reale fra più partizioni logiche in modo che tutte riescano a erogare lavoro come nell'esempio citato, sono necessarie delle tecniche di clustering estrememente sofisticate. Attualmente solo il Parallel SYSPLEX dello z/OS si avvicina a questi requisiti.

Di fatto si è arrivati a stabilire un criterio empirico per cui una LPAR in ambiente Mainframe e z/OS non dovrebbe gestire più di 6-7000 MIPS; questo obbliga a partizionare logicamente anche se oggi una singola istanza di z/OS o z/VM è in grado di gestire fino a 64 processori.

3.0 Il Sistema Operativo z/OS

Questo capitolo descrive le più importanti caratteristiche del Sistema Operativo IBM denominato z/OS; si tratta del Sistema Operativo maggiormente diffuso sui Sistemi Centrali di z/Architecture.

3.0.0 Introduzione

Il Sistema Operativo z/OS è il sistema che meglio utilizza le caratteristiche offerte dalla z/Architecture. Esso rappresenta l'ultimo gradino di una evoluzione le cui tappe fondamentali sono OS/360 (1964), MVS - Multiple Virtual Storage (1974) e OS/390 (1996). La versione corrente è la 1.11.
z/OS è sviluppato nei laboratori di Poughkeepsye (NY) da IBM Corporation e viene concesso in licenza d'uso agli utenti. Esso rappresenta il Sistema Operativo maggiormente usato dagli utenti di mainframe nel mondo. A tutt'oggi è il sistema che gestisce la massima parte delle applicazioni "Mission Critical" di molte grandi organizzazioni, specialmente di grandi Banche e grandi Enti Governativi.
In Italia è utilizzato da tutte le maggiori Banche e da molti Enti Centrali presso i quali gestisce le applicazioni Core Business.
Le ragioni del successo di un tale Sistema Operativo e la sua lunga permanenza sul mercato (le prime versioni risalgono al 1964) vanno ricercate in due importanti caratteristiche:

1. il fatto che esso sia stato "pensato" e quindi progettato "ab initio" come gestore di grandi volumi di dati e transazioni con molti utenti concorrenti con alti livelli di di integrità e sicurezza;
2. la caratteristica di compatibilità verso l'alto delle applicazioni che sotto il suo controllo vengono eseguite: è molto frequente che utenti di z/OS eseguano ancora oggi senza modifiche applicazioni progettate negli anni '60 ed ancora valide dal punto di vista applicativo.

Tuttavia negli ultimi anni z/OS è profondamente mutato per adeguarsi alla crescita dell'hardware, alle nuove implementazioni dell'Architettura e per aderire ai nuovi standard dell'informatica quali ad esempio il linguaggio C prima e Java dopo, i protocolli TCP/IP, gli standard web html/http. Tutte queste tecnologie emergenti ed i relativi "standard" fanno parte oggi del Sistema Operativo z/OS, che comunque rimane compatibile con gli ambienti tradizionali.
Tra le sue caratteristiche è molto importante la capacità di realizzare Cluster di Sistemi anche distanti fra di loro parecchie decine di chilometri (denominati Parallel Sysplex o Geographically Dispersed Parallel Sysplex per le soluzioni di Disaster Recovery), che consentono continuità operativa dell'infrastruttura anche a fronte di interruzioni di uno dei suoi nodi (Business Continuity). La continuità viene realizzata attraverso una completa condivisione dei dati e senza perdita di alcuna transazione. Una continuità di questo tipo richiede la capacità da parte della piattaforma di ridistribuire il carico fra i nodi superstiti ed eventualmente attivare la capacità di backup ("on demand").
Tale versatilità permette di aumentare o diminuire dinamicamente e senza interruzione di servizio la capacità di calcolo dei Sistemi, il numero e la velocità dei

processori che lo compongono, il numero ed il tipo di risorse di I/O collegate. Questa caratteristica, presente nell'hardware, viene utilizzata dai Sistemi Operativi della famiglia "mainframe" da sempre e solo nel recente passato ha iniziato a comparire sul mercato in altri sistemi basati su altre architetture e sistemi operativi.
Di seguito vengono riportate le caratteristiche principali del Sistema e le sue componenti:

- z/OS è un Sistema Operativo a 64 bit
- è stato progettato per trattare grandi volumi di lavoro
- è concepito per servire contemporaneamente molti utenti
- è ottimizzato per gestire grandi volumi di dati
- è progettato per servire applicazioni "critiche" per il business in maniera sicura

Componenti dello z/OS

Sono parte essenziale dello z/OS le seguenti funzioni:

- Dispatcher: il componente che manda in esecuzione i processi sui CP a disposizione.
- Sicurezza ed autenticazione, attraverso meccanismi specifici o standard "di fatto" come LDAP. Si veda il capitolo dedicato alla sicurezza.
- Gestione del collegamento alla rete secondo gli standard più diffusi (ad esempio TCP/IP o SNA);
- Programmi per gestire i dispositivi di I/O;
- Programmi per il controllo del Sistema;
- Programmi di Monitoring e misura del Sistema;
- Gestione della memoria virtuale
- Gestione della memoria reale
- Caricamento dei programmi eseguibili in memoria virtuale e loro condivisione fra i processi (Content management)
- Funzioni di gestione dei sistemi storage (System Managed Storage)
- Meccanismi automatici per la ripartenza;
- Programmi di gestione del Cluster (Parallel Sysplex);
- Programmi di Utilità e Servizio.
- Workload management

Lo z/OS è inoltre integrato da una serie di prodotti quali:

- Compilatori per linguaggi Evoluti;
- Data Base Managers:
- Security Manager and Auditing;
- Web Servers;
- Java Application Servers;
- Java Virtual Machine;
- XML Processing functions;
- Message Queuing and Routing Functions;
- Transaction Managers;
- Altri prodotti sviluppati da terze parti: Independent SW Vendor (ISV).

3.0.1 L'Address Space

z/OS si basa sull'uso della Memoria Virtuale. In particolare in z/OS viene implementata la funzione delle memorie virtuali multiple. L'idea base è che ad ogni processo venga assegnato per proprio uso esclusivo uno spazio virtuale grande quanto l'indirizzamento permesso dall'architettura. Questi spazi virtuali vengono chiamati **Address Space (AS).**

L'AS come contenitore di processi

All'interno di ogni Address Space l'esecuzione vera e propria è identificata da due diverse strutture, denominate anche Unit of Work (UoW): Task Control Block (TCB) e Service Request Block (SRB); qualunque processo in esecuzione sul Sistema, è sempre riconducibile ad una serie di TCB o SRB i quali a loro volta indicano i programmi che debbono essere eseguiti per quel processo.

Le componenti eseguibili (programmi) di z/OS si distinguono in:

- Programmi (moduli) di **Sistema**, che sono le componenti del Sistema Operativo prodotte da IBM
- Programmi (moduli) dell' **Utente**

z/OS è progettato per essere eseguito su Sistemi a Processori Multipli Simmetrici (SMP). Non esistono "a priori" criteri di affinità tra un certo processore ed i TCB ed SRB attivi nel sistema. Infatti in linea di principio ogni UoW è eseguibile su qualunque processore a qualunque Address Space appartenga (vedi figura 1). L'introduzione dei processori specializzati ha creato alcune "affinità" nell'esecuzione:

- Ogni TCB, nel momento in cui esegue codice Java, può essere eseguito sui processori zAAP (zSeries Application Assist Processor).
- Alcuni SRB relativi alla gestione di dati relazionali (DB2) possono essere eseguiti sui processori zIIP (zSeries Information Integration Processors).

Quali criteri guidano il Dispatcher nella scelta della UoW da mandare in esecuzione sul CP? In assenza di Hyperdispatch (vedi avanti) lo schema è il seguente:

- Ogni UoW ha tre stati mutuamente esclusivi:
  - Ready (puo' usare i CP)
  - Wait (è in attesa di un evento – ad esempio il completamento di un'operazione di I/O).
  - Non-dispatchable (non può usare i CP)
- Tutte le UoW Ready sono raggruppate in **un'unica coda.**
- Queste UoW sono ordinate nella coda secondo la Dispatching priority dell'AS in cui risiedono.

- La dispatching priority è un valore (da 0 a 255) gestito dal WLM (vedi piu'avanti). Maggiore il valore maggiore la dispatching priority.

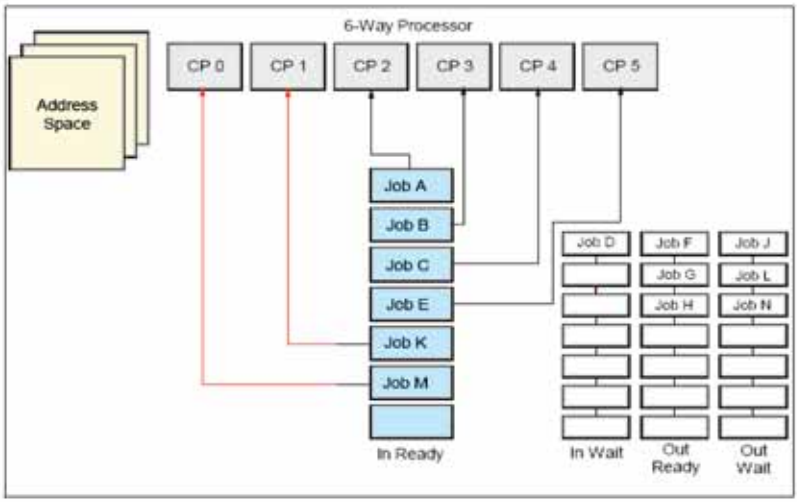


Figura 1 Dispatching in un sistema multiprocessore

Ogni volta che un evento (I/O interrupt, External Interrupt, etc) provoca l'intervento del dispatcher questo esamina lo situazione delle UoW Ready: se e' in coda una UoW che ha una dispatching priority più alta di una che è in esecuzione su un CP quest'ultima viene sospesa (preempted), messa in coda, e la UoW a più alta priorità mandata in escuzione sul CP liberato.

Analogamente quando lo stato di una UoW cambia da Ready a un altro dei due stati (perchè ha iniziato un'operazione di I/O o perchè il WLM ha deciso di sospenderla temporaneamente dall'accesso ai CP), il CP su sui era in esecuzione si libera e la UoW in coda a più alta priorità viene mandata su questo CP.

Per queste sue caratteristiche il Dispatcher dello z/OS è definito Preempting Dispatcher. Non è previsto un minimo garantito di uso dei CP da parte delle UoW (Time slicing).

L'AS e la memoria

Lo z/OS ha scelto per l'implementazione della memoria virtuale una dimensione della pagina virtuale di 4KB (4096 Byte). Ogni Address Space è suddiviso in Pagine, Segmenti, Regioni esattamente secondo la definizione architetturale. La dimensione della pagina virtuale determina le dimensione della Frame reale e dello Slot ausiliario: tutti avranno una dimensione uguale pari a 4 KB.

La traduzione dinamica degli indirizzi viene eseguita dal dispositivo hardware denominato DAT (Dynamic Address Translator) descritto nella sezione dell'architettura. Le Pagine della memoria centrale prendono il nome di Frames, quelle della memoria ausiliaria prendono il nome di Slot. Nella figura 2 si vede la relazione fra Memoria Virtuale, Memoria Reale e Memoria Ausiliaria.

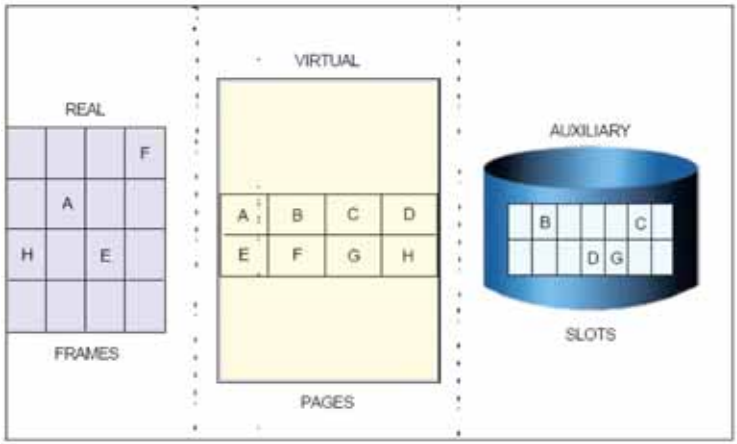


Figura 2 Memoria virtuale, reale, ausiliaria

Abbiamo un oggetto SW che occupa 8 pagine (da 'a' ad 'h'). Si noti come solo in memoria virtuale l'oggetto esista nella sua interezza; su memoria reale o ausiliaria esisteranno solo copie di alcune pagine (secondo le esigenze della gestione ottimale della risorsa memoria reale). L'esigenza della contiguità fra le parti è soddisfatta solo nella memoria virtuale. Inoltre una pagina virtuale avrà sempre una copia o in una frame o in uno slot o su entrambi.

Per supportare questa funzione un componente dello z/OS, il Real Storage Manager (RSM) provvede a costruire Region Table, Segment Table e Page Table. In z/OS viene assegnato un Address Space (AS) ad ogni processo. Ogni utente che si connette ad un sistema z/OS via TCP/IP è un processo autonomo che risiede in un proprio AS. Per ogni AS esiste un set di Region/Segment/Page Table che serve alla traduzione degli indirizzi di quell'AS.

Un AS è una struttura "logica", quindi esso avrà le dimensioni massime indirizzabili dall'Architettura. Essendo la z/Architecture un'architettura a 64 bit, la dimensione dell'AS sarà di 16 Exabyte (2**64). Poiché la dimensione dell'Address Space è aumentata continuamente, l'architettura supporta tre modalità di indirizzamento (24, 31 e 64 bit).

Perciò all'interno di un AS si individuano due confini ben precisi indicati nella figura 3.

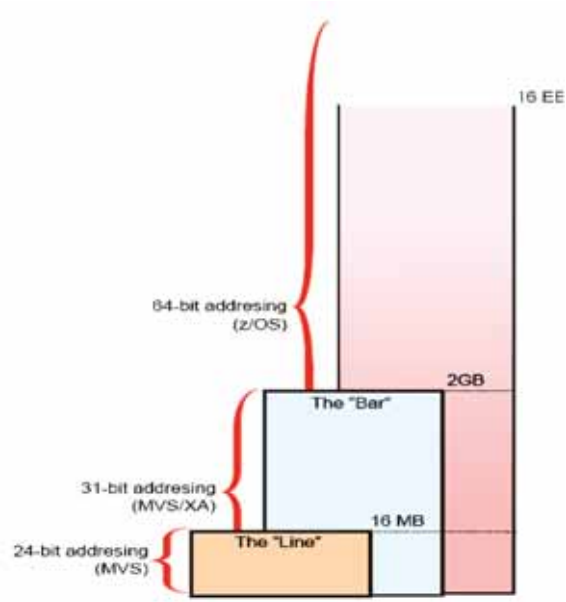


Figura 3 Differenti memory addressing

La linea dei 16 Megabytes (indicata anche come ***The "Line"***) per programmi che funzionano con indirizzamento a 24 bit. Questi programmi dovranno risiedere nella zona compresa 0 e 16 MB (cioè risiedere ***below the line***).
La linea dei 2GB (indicata anche come ***The "Bar"***), al di sotto della quale devono risiedere i programmi a 31 bit.
La zona al di sopra dei 4GB (e fino a 16 Exabyte) a disposizione per tutte le applicazioni in grado di sfruttare l'indirizzamento a 64 bit.

I programmi che supportano solamente indirizzi a 31 bit dovranno risiedere nell'area da 0 a 2GB. La parte di AS fra 2 e 4 GB è resa indisponibile per garantire l'integrità delle elaborazioni quando due programmi operanti in addressing mode differenti (31 o 64 bit; il problema non si pone con 24 bit mode) si passano degli indirizzi di memoria.
Ad oggi lo z/OS supporta l'indirizzamento a 64 bit solo per Operand Fetch e non per Instruction Fetch. In altre parole potranno risiedere *above the bar* solo dati; i programmi in esecuzione dovranno risiedere sotto i 2GB (l'instruction fetch di fatto sarà a 31 bit anche se si è in 64 bit mode). In altre parole: lo spazio sopra i 4 GB è disponibile solo per dati. E' una limitazione che non preoccupa; vige il noto aforisma: ci vogliono parecchie settimane/uomo per riempire un MB (memoria virtuale) di programmi eseguibili (bisogna scriverli) mentre bastano pochi secondi per saturare parecchi GB di dati (basta cambiare un parametro di bufferizzazione di un Data Base Manager). Infatti gli unici casi in cui oggi un AS da 2GB rischia la saturazione è quando ospita i buffer pool dei DB manager.

D'altro canto posizionare i programmi sopra i 4 GB virtuali avrebbe conseguenze pesantissime sulla riscrittura del sistema operativo se si vuole garantire la compatibilita' all'indietro (cioè con un mondo applicativo basato sui 31 bit).

La Figura 4 indica la struttura di un Address Spaces per un Sistema a 64 bit. All'interno dell'AS rileviamo la presenza di parti "comuni", "shared" e "private".
La parte comune di un AS ha questa caratteristica: il suo contenuto e' uguale in **tutti** gli AS esistenti nel sistema.
Viceversa il contenuto della parte privata può variare da un AS all'altro. Da questo punto di vista la parte "shared" si considera privata.
Le conseguenze sulla gestione della memoria reale sono che una pagina di memoria virtuale dell'area comune avra' bisogno solo di una Frame, qualunque sia il numero di AS presenti nel sistema (e questo fatto viene addirittura considerato nella gestione del Translation Lookaside Buffer (v. Architettura). La parte comune e' quindi il luogo ideale per posizionare programmi e dati che si vuole siano accessibili da tutti gli AS.
Non esiste parte comune sopra i 4 GB.
Esiste una parte sopra i 4 GB definita "shared". In quest'area vengono allocati tutte le aree di memoria ("memory object") che si vuole siano accedute da processi residenti in AS diversi.
Si osservi che dire che tutti gli AS hanno lo stesso modulo "comune" non vuole dire che esso venga copiato tante volte quante sono gli AS, bensì che esso viene "usato" da tutti e quindi fa idealmente parte di tutti gli AS.
Le parti "**private**" sono invece tipiche di un certo processo e solo di quello: esse contengono ad esempio i programmi ed i dati utilizzati da un certo utente del Sistema Operativo in un certo momento.
Si osservi che se lo stesso programma viene utilizzato da due utenti contemporaneamente (possibile a certe condizioni che vedremo più avanti), esso sarà associato a due processi diversi.

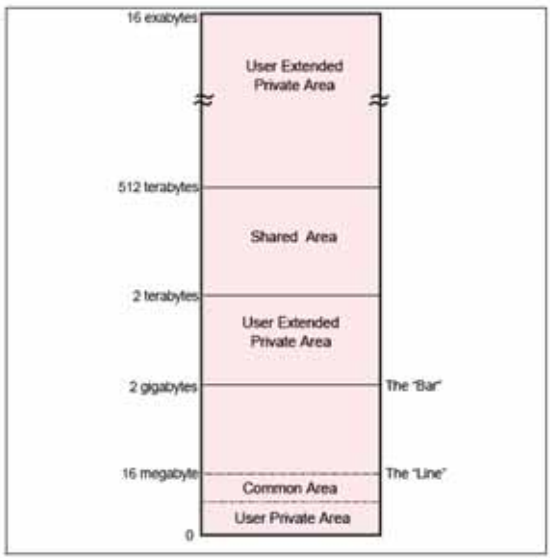


Figura 4 Struttura di un Address Space

La Figura 5 mostra il dettaglio della struttura di un AS a 64bit e di questo evidenzia le parti comuni (Common), Private e Condivise (Shared).
Osserviamo la parte comune:

- Nucleus è la parte dello z/OS che viene caricata alla partenza della partizione logica (Initial Program Load – IPL). Si divide in due porzioni, sopra e sottto la linea dei 16MB per tener conto di quelle parti ancora scritte a 24 bit (ormai praticamente sparite). La memoria virtuale relativa è fissata in memoria reale (FIXED virtual storage); cioè anche se l'esecuzione del codice del nucleo è DAT ON, le sue pagine risiedono sempre in memoria reale, e l'associazione pagina-frame non cambierà se non al prossimo IPL.

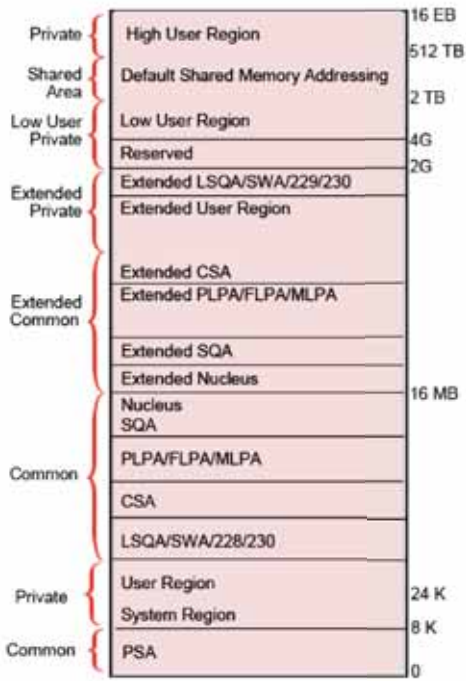


Figura 5 Struttura di un Address Space (dettaglio)

- PLPA/FLPA/MLPA (Pageable/Fixed/Modified Link Pack Area) è un' area in cui vengono caricati programmi usati da tutti gli AS. Quest'area è paginabile, e ha un Data set di page dedicato (PLPA Page Data Set).
- System Queue Area (SQA) e' un'area in cui risiedono dati accessibili a tutti gli AS; come per Nucleus quest'area e' fissata in memoria reale (cioe' non è paginabile)
- Common Service Area (CSA) ha la stessa funzione della SQA senza il requisito FIXED; può essere paginata e ha un Data set di Page dedicato (Common Page Data Set).

Queste quattro aree hanno una parte sotto la linea dei 16 MB e una parte sopra la linea dei 16 MB: questa è una conseguenza del requisito di compatibilità all'indietro del mondo Mainframe. Poiché la logica degli AS si estende a tutti i processi (programmi) presenti nel Sistema, cioé anche ai moduli del Sistema Operativo stesso, distinguiamo diversi tipi di AS:

- **System Address Spaces**: vengono creati all'avvio del Sistema Operativo (IPL - Initial Program Load) ed eseguono tutte le funzioni necessarie al funzionamento del Sistema e degli altri AS.

- **SubSystem Address Spaces**: Vi risiedono i sottosistemi. Essi sono un insieme di funzioni a disposizione delle applicazioni che possono richiedere i servizi che i sottosistemi offrono con una particolare interfaccia: la **Subsystem Interface.** Sono sottosistemi ad esempio i Gestori di Basi di Dati (DBMS) o alcuni Middleware o il JES2.
- **TSO/E Address Spaces:** Il TSO/E è un'applicazione che consente l'accesso al Sistema z/OS da parte di utenti che dispongono di un terminale (o di un PC con emulazione 3270). Per ciascun utente che accede al Sistema il TSO/E crea un singolo Address Spaces di questo tipo.
- **User Address Spaces**: creati per ogni programma che viene eseguito nel Sistema al di fuori da tutti i casi precedenti (per esempio un programma Batch, cioé senza interventi umani continuativi).

Il numero di AS attivi in un Sistema z/OS è quindi grande, mediamente dell'ordine delle centinaia; è bene osservare che il numero di AS attivi non è direttamente correlato al numero di utenti che stanno interagendo col Sistema in un certo momento: se è vero infatti che per ogni utente che accede tramite TSO/E viene creato un address space, è anche vero che esistono dei Sottosistemi transazionali (vedi dopo) che a fronte di un solo o di un numero contenuto di AS sono in grado di servire decine di migliaia di utenti.
Poiché un AS è associato ad un utente, l'AS diventa anche il contenitore per tutti i processi che la presenza dell'utente innesca nel sistema.

Il meccanismo degli AS è uno strumento molto potente per aumentare la integrità delle elaborazioni. Poichè ogni processo ha un AS grande quanto l'indirizzamento permette, risulta molto difficile che altri processi siano alterati a causa di malfunzioni che si verificano in altri processi/AS presenti nello stesso sistema.

3.0.2 La gestione della Memoria Reale

Le risorse Hardware su cui insiste l'implementazione della memoria virtuale dello z/OS sono:

- Memoria Centrale – ovvero la RAM a disposizione dei processi ed accessibile dai Processori;
- Memoria Ausiliaria – ovvero qualsiasi altro dispositivo in grado di ospitare temporaneamente i contenuti della memoria Centrale quando essa non riesce più a contenerli.

Perché un programma possa essere eseguito è necessario che le pagine virtuali che contengono le istruzioni in corso di esecuzione e gli operandi che esse richiedono si trovino in memoria reale. Tutte le altre pagine virtuali relative al programma possono trovarsi sulla memoria ausiliaria. Quindi la Memoria Reale necessaria all'esecuzione di un programma può essere minore della dimensione del programma stesso (vedi figura 6).

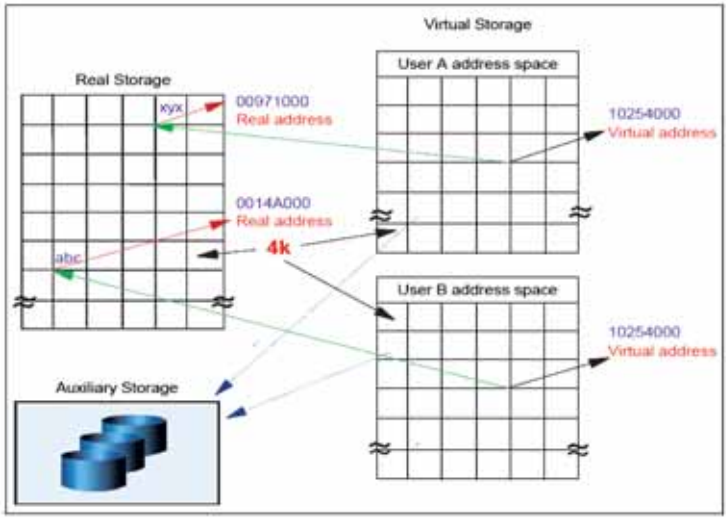


Figura 6 Memoria virtuale e Memoria reale

Cosa avviene quando l'esecuzione di una istruzione non trova in memoria reale i campi di cui ha bisogno? o, in termini più aderenti all'architettura, cosa avviene quando una o più pagine contenenti i campi necessari all'esecuzione dell'istruzione non si trovano in memoria reale? Questo evento viene indicato dall'architettura con il termine **Page Fault**.
Ogni sistema operativo costruito sulla z/Architecture deve provvedere a portare in memoria reale da disco la pagina mancante: deve cioé effettuare l'operazione di I/O che trasferisce 4096 byte (il contenuto della pagina mancante) dai **Page Data Set** (su disco) in Frame in memoria reale. Questa operazione si chiama **Page-In** e ha la durata tipica di ogni operazione di I/O su disco: tipicamente 1 msec. L'esecuzione di un Page-In è sincrona rispetto all'esecuzione dell'istruzione che l'ha causato, o, in altre parole, l'esecuzione dell'istruzione è sospesa finchè il Page-In non è completato.
E' facile fare il confronto fra l'esecuzione vera e propria dell'istruzione, pochi nanosecondi, e l'esecuzione dell'operazione di I/O che risolve un Page Fault: il rapporto è vicino al milione di volte. E' evidente come sia indispensabile per garantire un'efficiente esecuzione di un programma che gli eventi di Page Fault siano tenuti al minimo, senza ricorrere all'ovvia soluzione che tutto il programma stia in memoria reale perché così si avrebbe bisogno di una quantità di memoria reale molto più grande di quella sufficiente per avere delle performance accettabili.
Introduciamo il concetto di Working Set di un programma: è l'insieme delle Frame Reali che ospitano le pagine virtuali relative al programma. Non è detto che tutte le sue pagine virtuali siano residenti su Frame reali; anzi, la condizione normale è che il Working set sia inferiore alla dimensione virtuale del programma.
Per quanto detto precedentemente, poiché per eseguire un programma non è necessario che esso debba risiedere tutto in memoria reale, esiste una dimensione di Working Set ottimale, inferiore a quella del programma, che consente di eseguire il programma stesso con una penalizzazione accettabile (causata da qualche Page-In

necessario per risolvere i Page Fault). Un certo numero di Page-In durante l'esecuzione di un programma è da considerarsi fisiologico. E' perciò essenziale avere dei meccanismi di controllo che garantiscano ad ogni programma di avere il proprio Working Set ottimale. Una delle funzioni essenziali del Workload Manager è appunto di garantire questo stato per ogni processo presente nel sistema. La corretta gestione della Memoria Centrale dovrà perciò garantire a tutti i processi attivi nel sistema un working set ottimale.

Nello z/OS questo viene ottenuto da due componenti:

- Real Storage Manager (RSM).
- System resources manager (SRM)

Questi due componenti cooperano per:

- Garantire che sia sempre disponibile un pool di Frame libere (Available Frame Queue - AFQ) per soddisfare i Page-In. Queste frame sono libere in quanto non sono associate a nessuna pagina.
- Determinare quali Frame possano essere liberate del loro contenuto virtuale per poter essere assegnate al pool AFQ. Questa attività è estremamente critica. Infatti liberare una frame implica che il working set di qualche programma diminuisce. Bisogna adottare una strategia che mantenga AFQ ad un livello accettabile; lo z/OS monitorizza costantemente questo livello e quando questo scende sotto una certa soglia innesca il processo di ripristino del livello accettabile attraverso l'RSM – questa attività si chiama AFQ replenishment nella terminologia z/OS - senza penalizzare eccessivamente i working set dei processi attivi nel sistema. Un effetto collaterale del liberare una frame dal suo contenuto virtuale è che questo contenuto non deve essere perso; perciò esiste il processo di **Page-Out** che copia il contenuto della frame su un Page Data Set. Attenzione: il Page out sarà fatto solo se la pagina è stata modificata dall'ultimo page in; altrimenti la frame relativa verrà semplicemente mossa nell'AFQ. Quando l'operazione di I/O collegata al Page out sarà conclusa, la frame sarà considerata libera e mossa nell'AFQ.

Ogni volta che il processo AFQ replenishment deve essere eseguito, il componente System Resource Manager (SRM) dello z/OS viene innescato. L'SRM interfaccia con RSM per portare a termine l'attività. Le Frame da liberare sono scelte secondo questi criteri:

- Least Recently Used (LRU) - le Frame inattive da più tempo vengono liberate
- Parere del Workload Manager - per garantire i livelli di servizio si deve garantire alle applicazioni un Working Set adeguato
- Frame modificata o no - si tende ad evitare il Page Out.

L'attività che sottrae Frame al Working Set di un Address Space è il "**Page Stealing**". Il meccanismo LRU viene usato perché si assume che le Frame inattive da più tempo siano quelle che hanno minore probabilità di essere referenziate in futuro. Si suggerisce al lettore una meditazione sull'effetto che il meccanismo di Page Stealing ha sulla gestione HW del Translation Loookaside Buffer.

Per la corretta gestione del meccanismo LRU lo z/OS mantiene un contatore per ogni Frame che indica da quanto tempo la Frame non viene acceduta (Unreferenced Interval Count – UIC); per gestire questo contatore lo z/OS sfrutta il Reference bit (v. capitolo sull'architettura). Questo bit viene messo ON ogni volta che la Frame viene referenziata. L'analisi dei Reference bit viene fatta dal System Resource Manager (SRM) secondo intervalli prefissati. Un intervallo tipico di quest'analisi è di qualche secondo.
Gli UIC (uno per Frame reale) si trovano in una struttura chiamata Page Frame Table (PFT) che contiene un'entrata (Page Frame Table Entry – PFTE) per ogni Frame Reale disponibile allo z/OS.
Poiché liberare delle Frame implica eseguire operazioni di I/O (per il Page-Out), è possibile evitare questa attività sapendo se la Frame è stata modificata. Il Change Bit (v. Architettura) assiste in questo. Esso diventa ON ogni volta che la Frame viene modificata. Se si deve liberare una Frame il cui Change bit è OFF la Frame è semplicemente mossa nell'AFQ senza Page-Out (non è stata modificata dall'ultimo Page-Out); altrimenti ne viene eseguito il Page-Out e, terminata l'operazione di I/O relativa, il corrispondente Change bit viene messo OFF.

Il complesso di Page-In + Page-Out viene indicato come Paginazione. Con l'aumentare delle dimensioni delle Memorie Centrali e con il loro minore costo, l'entita' della paginazione e' costantemente diminuita. Va detto però che il processo di Paginazione per il Sistema z/OS deve essere ritenuto "fisiologico" in quanto facente parte della struttura stessa del Sistema e pertanto esso è sempre presente anche se in piccola parte.

Per riassumere diremo che i Page-In sono sincroni all'esecuzione del programma, quindi il tempo speso sarà parte integrante dell'Elapsed Time del programma coinvolto. Viceversa l'attività di Page Stealing e i possibili Page-Out collegati sono asincroni rispetto all'esecuzione dei programmi e l'unico effetto su di loro sarà la conseguenza della riduzione del Working Set; sarà cura del WLM guidare il page stealing in modo da riuscire a soddisfare sempre i goal stabiliti dall'installazione.

Un meccanismo analogo alla Paginazione, ma di portata diversa, prende il nome di "Swapping".
Può avvenire che un AS resti inattivo per parecchio tempo (qualche secondo). Questa condizione (Long wait – esattamente codificata all'interno dello z/OS) è tipica di un utente interattivo (terminale) che, fra un'interazione e la successiva non opera con il Server poiché "sta pensando" (questo intervallo viene spesso indicato con Think Time). I Think Time hanno dimensioni "umane" e sono tipicamente dell'ordine di decine di secondi. Perciò quando un'interazione è terminata (è terminata una transazione) l'Address Space relativo all'utente viene Swapped-Out in quanto il sistema immagina che l'utente non genererà transazioni per parecchi secondi. Quindi quando l'utente immetterà un nuovo comando il sistema provvederà a fare lo Swap in dell'AS relativo.
Lo Swap out è eseguito in due fasi:

- L'AS viene dichiarato non eleggibile per il Dispatching (cioè nessun processo al suo interno può usare i CP)
- Opzionalmente tutte le frame che costituiscono il suo working set vengono rese disponibili. Si possono definire dei data set di paginazione specializzati (Swap data set) per gestire questo traffico di I/O; le operazioni di I/O verso questi data set sono ottimizzate per l'attività di Swap out e Swap in.

Oggi lo Swap-Out di un utente è prevalentemente solo logico poiche' la dimensione media della memoria reale configurata è aumentata notevolmente. Se lo Swap passa da Logico a Fisico allora il Working Set dell'Address Space verrà trasferito nell'AFQ.
L'unica conseguenza pratica di uno Swap-Out logico è che tutte le Unit of Work (UoW) attive nell'Address Space saranno rese non-Dispatchable (lo swap si limita alla prima fase).
Quando la condizione di Long wait cesserà (cioè l'utente interattivo avrà inviato un input al Server) l'AS sarà Swapped-In. Nel caso lo Swap-Out precedente fosse stato fisico avremo un'attività di Swap-In fisico (una serie di operazioni di I/O dai Page Data Set in Memoria Reale). Altrimenti lo Swap-In solo Logico si limiterà a ripristinare le UoW dell'AS come Dispatchable (possono nuovamente essere mandate in esecuzione su un Processore).

3.0.3 Interfacce Utente (User Interfaces)

Questo capitolo descrive le interfacce Utente verso il Sistema Operativo z/OS.

Un'interfaccia utente è lo strumento che consente di interagire col Sistema al fine di fornire comandi e verificarne il risultato. La richiesta di esecuzione di un programma può essere data attraverso un comando, come pure la richiesta di apertura o di cancellazione di un archivio.

Le interfacce utente alle quali siamo abituati nella pratica quotidiana sono ormai tutte di tipo grafico, tali interfacce sono presenti anche in z/OS, ma in effetti molto poco usate. Le interfacce presentate di seguito sono invece quelle più comunemente usate dai Programmatori di Sistema e dagli Operatori.

Tali interfacce non vanno confuse con le interfacce applicative, ovvero quelle attraverso cui gli utenti utilizzano le applicazioni eseguite sotto il controllo di z/OS. Ad esempio, la schermata attraverso la quale viene interrogato un conto bancario è un'interfaccia applicativa ed, in generale, essa non dipende dal Sistema Operativo del Server Centrale.

Occorre distinguere inanzitutto due tipi di dispositivo di comunicazione:

1. La Console di Sistema
2. Il Terminale video

La Console di Sistema **(System Console)** è costituita da uno o più dispositivi di I/O (generalmente video con tastiera) che rappresentano lo strumento di comunicazione primaria del Sistema. Attraverso la System Console, il Sistema fornisce già dall'avvio i messaggi informativi o di errore provenienti da tutti gli Address Spaces attivi e richiede le risposte ad eventuali situazioni nelle quali sia necessario assumere decisioni. Attraverso la System Console è possibile fornire direttamente comandi al Sistema Operativo o ai sottosistemi.

La console di Sistema è la prima interfaccia utente attivata da quando il sistema parte (Initial Program Load – IPL).

La System Console non è solo un dispositivo fisico, ma è anche una funzione logica. Per questa ragione essa può essere interrogata anche da un Terminale video sotto il controllo del TSO/E con appositi prodotti o può essere interfacciata da un programma automatico che funge da operatore.
In passato, in effetti, la Console era ordinariamente presidiata da un addetto, l'operatore di console, il quale aveva il compito di leggere i messaggi e prendere le relative decisioni. Oggi l'operatore di console viene sovente sostituito da un Software che opportunamente programmato è in grado di assumere le decisioni di routine o di segnalare ad un responsabile eventuali anomalie non gestite. Uno dei motivi di questa evoluzione è che il flusso di messaggi è oggi talmente elevato che un'operatore "umano" difficilmente riuscirebbe a reagire tempestivamente. Perciò questi Software provvedono a prendere le decisioni più "automatizzabili" lasciando all'operatore le decisioni di più lungo respiro (ad esempio il far partire una sequenza di Attività Batch). In un sistema z/OS si possono avere contemporaneamente molte console di Sistema, si può operare in modo che certi messaggi di errore o di segnalazione (ad esempio quelli provenienti da un certo sottosistema) siano inviati solo a determinate console. I sistemi in Cluster (Parallel Sysplex) possono condividere lo stesso gruppo di console.

Le console accettano comandi su due linee ed hanno uno scorrimento verso l'alto dei messaggi. I messaggi e le risposte sono memorizzati su un file (detto Syslog) e pertanto possono essere rivisualizzati a pagine (funzione di re-display), anche con opportuni filtri.

Il **Terminale Video** ha una funzione decisamente diversa essendo soggetto a maggiori controlli, ma ha funzioni molto più avanzate nella gestione del Sistema.

Attraverso il terminale video si può accedere con tre modalità diverse, ricordiamo ancora una volta che si sta qui discutendo di Operazioni di Amministrazione del Sistema e non di Applicazioni:

1. Modalità TSO (Time Sharing Option)
2. Modalità ISPF/PDF (Interactive System Programming Facility / Program Development Facility)
3. z/OS UNIX Shell

Per meglio comprendere la differenza tra le tre opzioni, è opportuno ricordare che la codifica usata nativamente dal Sistema Operativo z/OS è quella EBCDIC e che tale codifica prevede l'uso di terminali video denominati IBM 3270 (o semplicemente **3270**). Tali terminali prevedono un protocollo di trasmissione SDLC (VTAM) ed una modalità di gestione "a pagine". Una pagina è ordinariamente costituita da una schermata di 1920 (24 x 80) caratteri. La caratteristica principale della modalità e del protocollo 3270 è la possibilità di muoversi con il "cursore" all'interno della "pagina" e poi inviarne tutto il contenuto in una sola volta mediante la pressione di un tasto detto "invio" (o ENTER); tale funzione è particolarmente utile quando si fa l'Editing di testi o di codice sorgente di programmi.
L'interazione tra il Terminale ed il Sistema Operativo avviene solo alla pressione del tasto ENTER.

Negli ultimi dieci anni la modalità 3270, che rimane quella nativa e quindi più usata, è stata affiancata in z/OS ad un'altra proveniente dagli ambienti UNIX (e quindi dalla codifica ASCII) denominata Telnet (o VT100). Tali terminali sono caratterizzati dalla

codifica ASCII e sono spesso usati con modalità a linea o a carattere, essendo in questo caso l'interazione col Sistema Operativo molto più frequente. Le schermate sono caratterizzate da un riempimento dall'alto verso il basso con scorrimento verso l'alto.

Contemporaneamente a questa caratteristica è stata introdotta una modalità operativa denominata z/OS UNIX System Services (USS) che consente all'utente che si interfaccia col Sistema di comportarsi come se stesse colloquiando con una "Shell" UNIX. In questo caso le modalità di accesso, le organizzazioni degli archivi e le utilities utilizzate sono quelle definite dalle specifiche POSIX relative all'ambiente UNIX.

Nonostante la codifica intrinseca dei dati e delle istruzioni rimanga EBCDIC, l'interfaccia potrà utilizzare direttamente terminali con codifica ASCII (VT100 o Telnet).

Di seguito vengono descritte le tre interfacce.

Figura 7 Logon al TSO

Interfaccia TSO/E: rappresenta la più semplice ed antica interfaccia verso z/OS, quando un utente collega un terminale 3270 al Sistema Attivo, viene visualizzata una schermata detta di LOGON (vedi figura 7).
Questi terminali non esistono più. Ma la modalità di collegamento resta e viene realizzata da particolari SW che girano su client Windows/Linux e che "simulano" una sessione basata su terminali 3270 virtuali. A loro volta i client sono collegati mediante TCP/IP al server z/OS.

Una volta che l'utente è stato identificato mediante una password, gli viene proposta una "linea di comandi" attraverso la quale egli può dialogare col Sistema. La presentazione di una linea comandi in effetti corrisponde alla creazione di un Address Space dedicato a quell'utente. Le caratteristiche di accesso e le modalità di creazione del relativo Address Space possono essere modificate all'installazione del Sistema.

La modalità TSO/E richiede una conoscenza approfondita dei comandi e per tale ragione è utilizzata solo da utenti molto esperti. Più frequente è l'uso combinato delle capacità di creazione di un Address Space offerte dal TSO con l'ambiente ISPF/PDF che è invece di più facile utilizzo. La figura 8 mostra l'utilizzo della linea comandi TSO/E per richiedere attraverso il comando "time" l'ora al Sistema e la relativa risposta.

Figura 8 Linea Comandi TSO

Interfaccia ISPF/PDF: questa interfaccia si fonda ancora sul TSO/E ma è caratterizzata da videate con scelte multiple nelle quali è possibile l'uso del "mouse" e, a richiesta, esse possono avere una grafica avanzata.

Figura 9 Interfaccia ISPF

L'ambiente PDF fornisce un potente EDITOR di testi e di programmi a pagine ed una interfaccia avanzata verso il TSO per poter dare comandi al sistema; tali comandi sono filtrati sulla base delle autorizzazioni dell'utente controllate dal Security Manager del Sistema Operativo.

La Figura 9 mostra le caratteristiche di una schermata tipica di ISPF/PDF che rappresenta il Menu standard di un Sistema z/OS. Le schermate ISPF/PDF sono dotate di una funzione di aiuto in linea.

Figura 10 Interfaccia ISPF - Menu

Le schermate ISPF/PDF se visualizzate attraverso un Client con capacità grafiche possono essere visualizzate in formato grafico avanzato, come si vede in figura 10.

Usualmente l'interfaccia ISPF è utilizzata da tutti gli utenti del TSO/E. ISPF fornisce anche limitate funzioni applicative ed è programmabile attraverso un linguaggio procedurale denominato REXX: per tale ragione molte funzioni applicative di base, legate alla gestione ed amministrazione del Sistema, vengono realizzate sotto il controllo di ISPF. Esempi sono la creazione degli utenti, l'amministrazione dei dati, la gestione dei nastri magnetici, la gestione dei lavori Batch, la gestione di stampanti e delle linee di comunicazione.

Di fatto ISPF rappresenta il "cuore" dell'amministrazione del Sistema ed il metodo di accesso principale alle relative funzioni di gestione.

Interfaccia UNIX (UNIX Shell): l'interfaccia UNIX venne introdotta in z/OS a metà degli anni '90 (a quel tempo il Sistema Operativo si chiamava MVS/ESA). Nello stesso periodo il Sistema Operativo OS/390 Rel. 1.2 ottenne la certificazione UNIX95 secondo gli Standard internazionali.

Alla intefaccia UNIX si può accedere in due modalità diverse:

1. Nativamente, cioè attraverso Terminali ASCII su TCP/IP
2. In modo emulato attraverso TSO/ISPF/PDF da Terminali 3270 mediante i comandi:
   - OMVS – che fornisce accesso in modalità UNIX emulata su 3270
   - ISHELL – che fornisce una interfaccia a pagina molto vicina all'ISPF

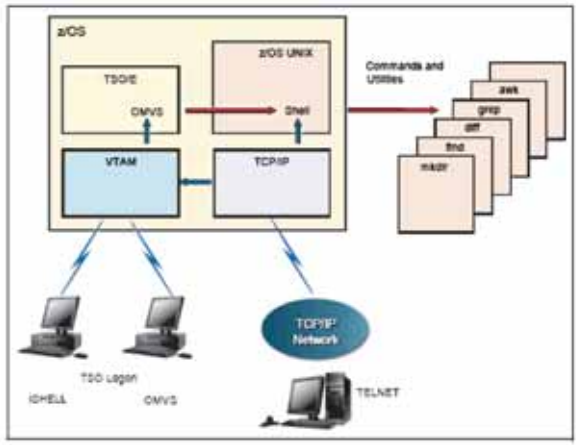


Figura 11 UNIX System Services

```
# Improve shell performance
if [ -z "$STEPLIB" ] && tty -s;
then
    export STEPLIB=none
    exec sh -L
fi
# Set the time zone as appropriate.
TZ=EST5EDT
# Set a default command path, including current working dir (CDW)
PATH=/bin:.
# Sets the path environment variables
LIBPATH=/lib:/usr/lib:.
MANPATH=/usr/man/%L
NLSPATH=/usr/lib/nls/msg/%L/%N
# Sets the language
LANG=C
# Sets the name of the system mail file and enables mail notification.
MAIL=/usr/mail/$LOGNAME
# Export the values so the system will have access to them.
export TZ PATH LIBPATH MANPATH NLSPATH MAIL LANG
# Set the default file creation mask - umask
umask 022
# Set the LOGNAME variable readonly so it is not accidentally modified.
readonly LOGNAME
```

Figura 12 Esempio di script UNIX

La Figura 11 mostra i due diversi modi e le tre diverse caratteristiche dell'accesso alla Shell UNIX di z/OS. Va ricordato che le modalità USS sono sensibilmente differenti da quelle TSO/E; infatti l'utente della Shell UNIX non ha un Address Space dedicato come gli utenti TSO, ma esegue i suoi lavori su un Address Space comune denominato OMVS, almeno fino a quando esegue comandi di Shell.

La Figura 12 mostra l'aspetto della Shell UNIX richiamata da un terminale ASCII di tipo VT100. La Shell UNIX può essere usata per richiamare script o per eseguire programmi (se scritti in maniera compatibile con l'ambiente UNIX). Su terminali ASCII l'unico editor disponibile è l'editor tradizionale degli ambienti UNIX denominato VI.

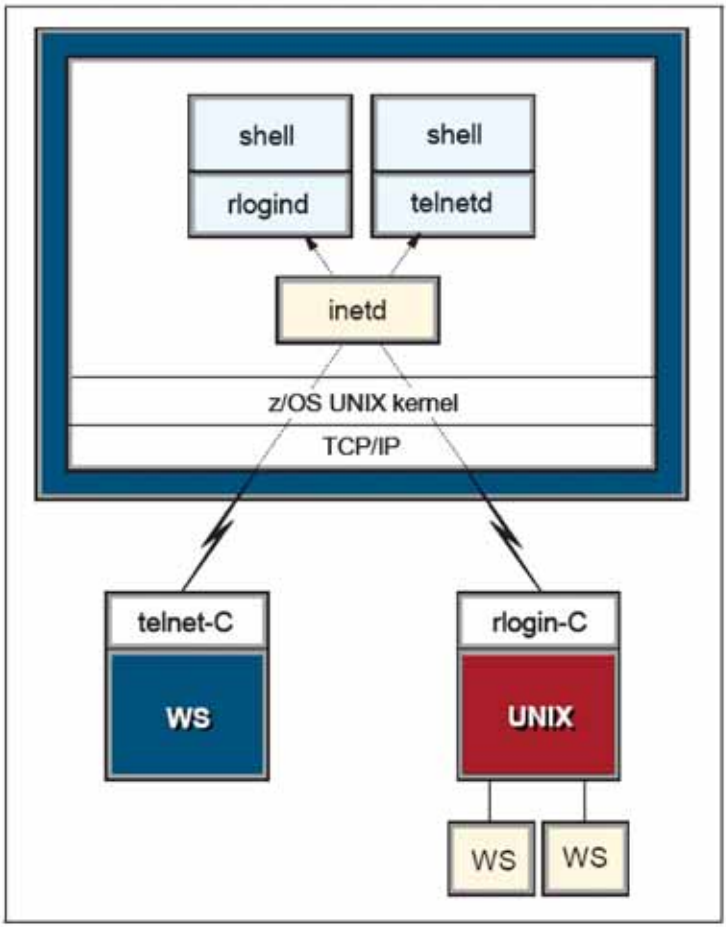


Figura 13 Modi per accedere alla shell UNIX

E' possibile connettersi all'ambiente USS di z/OS anche da altri Sistemi UNIX mediante la modalità di Remote Login o di Remote Telnet (Figura 13) .

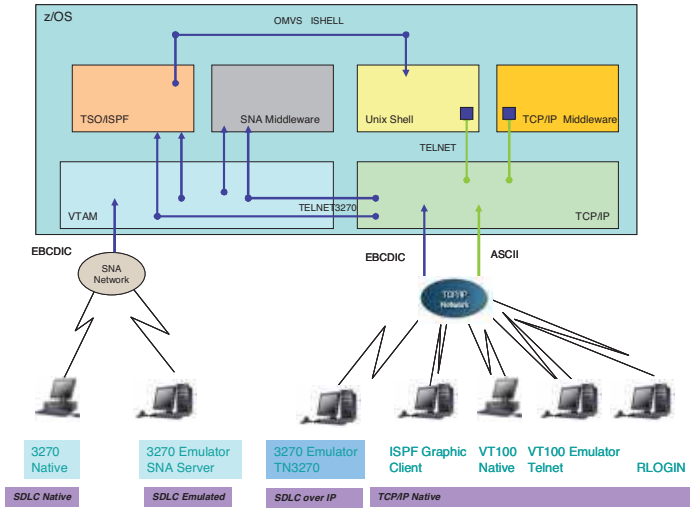


Figura 14 Le differenti interfacce verso z/OS

La Figura 14 sintetizza e riepiloga tutte le interfacce Utente disponibili per collegarsi ad un Sistema z/OS (non sono considerate le interfacce applicative).

3.0.4 Il JES2 (Job Entry Subsystem)

L'esecuzione di lavori "batch" ovvero "a blocchi" rappresenta ancora oggi una delle attività più importanti del Sistema Operativo z/OS.

Un lavoro batch è costituito da uno o più programmi eseguiti secondo un ordine prefissato e con scarsa o nulla interazione utente. In linea di massima un lavoro "batch" non richiede la presenza o l'interazione di un Terminale Video: il lavoro viene "sottomesso" al sistema attraverso diversi possibili meccanismi di input (il TSO/E o meccanismi automatici di schedulazione sono i mezzi più usati). Al termine dell'esecuzione vengono controllati i risultati prodotti (via TSO/E o con altri mezzi, ad esempio una stampa di controllo).

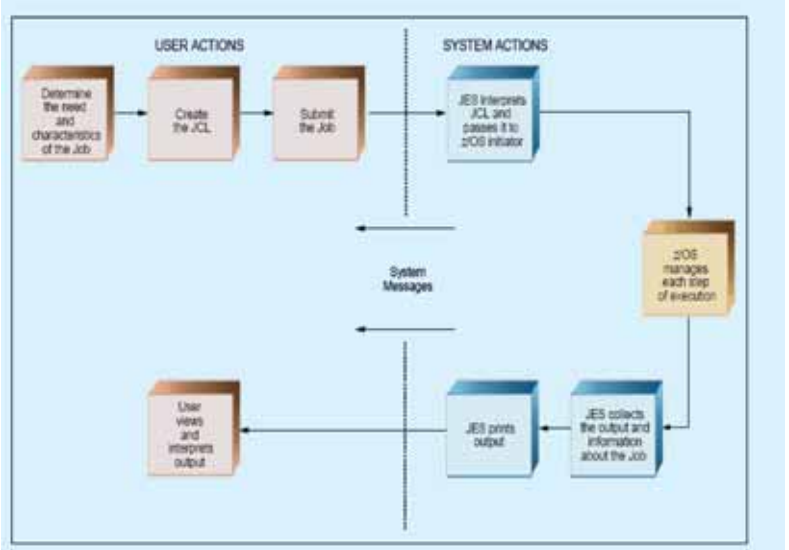


Figura 15 Elaborazione batch con JES

Per lungo tempo e nei primi anni dell'informatica, le eleborazioni "batch" hanno costituito l'unica modalità per eseguire lavori su un Calcolatore. Ancora oggi i lavori batch rimangono il modo di operare di molti processi applicativi: un lavoro batch è, ad esempio, la produzione dei cedolini dello stipendio dei dipendenti, la chiusura trimestrale dei conti correnti bancari o la stampa della lista dei passeggeri di un volo. E' chiaro che tutti questi lavori possono essere svolti anche con l'interazione diretta di un terminale e, in generale, il loro modo di operare dipende dalle scelte operate dal programmatore dell'applicazione. Nel testo si intendono lavori "batch" quei lavori nei quali l'interazione con l'uomo è minima, ovvero quei lavori che vengono "sottomessi" al Sistema e poi "controllati" al termine della loro esecuzione.

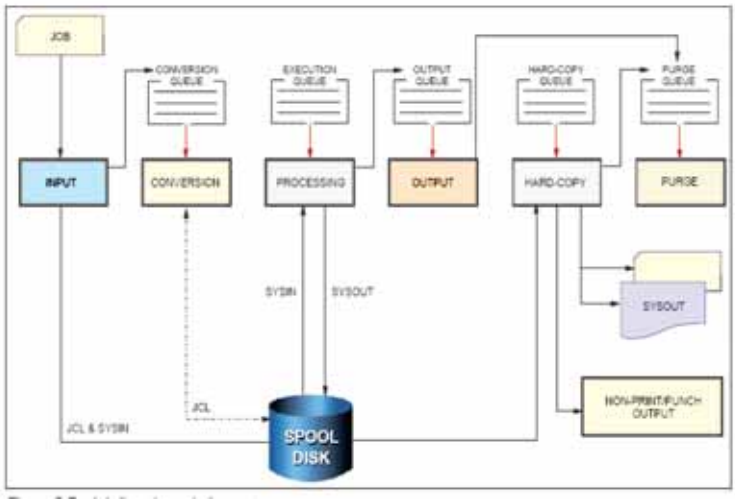


Figura 16 SYSIN e SYSOUT

La Figura 15 indica il flusso logico di un lavoro "batch" all'interno del Sistema Operativo z/OS. Vista la particolare importanza assunta storicamente dai lavori di tipo batch all'interno del Sistema Operativo, sono state specializzate delle componenti specifiche per la loro gestione: i Job Entry Subsystem (JES): la versione più usata di questi componenti si chiama Job Entry Subsystem 2 (JES2).
Il JES2 fornisce il supporto per la sottomissione, l'esecuzione e il controllo dei lavori di tipo batch. Esso consente, inoltre, la verifica ed il controllo dei risultati di tali lavori. JES2 fornisce anche un supporto generico per l'automazione dell'ambiente, ovvero per la creazione di appositi software in grado di automatizzare la gestione dei lavori batch; tali software prendono il nome di schedulatori (job scheduler).
Nel seguito indicheremo l'insieme dei programmi che costituiscono un lavoro batch col termine di "Batch Job" o semplicemente di "Job". Un job può contenere oltre ai programmi da eseguire anche alcuni dati di controllo e le istruzioni da sottomettere al JES2 affinché possa operare: per tali istruzioni è stato predisposto un apposito linguaggio procedurale denominato Job Control Language (JCL) che descriveremo in seguito.

Il JES2 deve essere attivato obbligatoriamente all'avvio del Sistema Operativo z/OS in quanto fornisce anche funzioni di base per l'esecuzione degli altri lavori, anche se non ti tipo batch; esso fornisce, ad esempio, il supporto per l'attivazione di altri sottosistemi o Middleware che in generale non sono correlati ai lavori di tipo batch.

La Figura 16 descrive il flusso operativo e le componenti del JES2, oltre al già citato Job Control Language che costituisce la via primaria per fornire dati al JES2; il diagramma riporta alcune fasi essenziali e le relative componenti proprie del Sistema Operativo che vengono invocate per gestire il Job:

- **Input:** Rappresenta la fase iniziale di sottomissione del JOB a partire dalle istruzioni in JCL. La sottomissione può avvenire attraverso TSO/E, attraverso un programma (Schedulatore Automatico), da un altro Job in esecuzione o con una particolare modalità definita "procedura", descritta nel seguito. Il JCL che descrive un job è contenuto in un file che sta in un archivio memorizzato su dischi magnetici con una struttura che ricorda il primo supporto che ha ospitato il JCL: la scheda perforata. Infatti gli statement JCL hanno una lunghezza fissa di 80 byte, tanti quanti erano i caratteri di una scheda perforata. Il componente usato è l'Input reader.
- **Conversion:** E' una fase molto importante del processo. Un apposito modulo del JES2 invoca una funzione chiamata Converter/Interpreter (C/I) il quale legge il file JCL che descrive il Job, lo interpreta, lo traduce in strutture dati che serviranno alla funzione seguente (Initiator) per lanciare l'esecuzione del Job, e accoda il Job nella Execution Queue; in questa fase il C/I verifica anche che nella stesura degli statement non ci siano errori di sintassi JCL. Un Job Batch ha bisogno di un ambiente di esecuzione: ad esempio, se si vuole stampare la lista dei passeggeri di un volo occorre disporre di un archivio (File di dati) contenente i dati dei passeggeri, di un programma di stampa e di una stampante. I riferimenti all'ambiente necessario al programma batch, ad esempio, il nome del File di dati, il nome del programma, sono definiti nel JCL e vengono controllati nella fase di conversione. Se uno degli elementi richiesti non viene trovato si ha un errore (JCL error) ed il processo batch viene interrotto.
- **Processing:** E' la vera fase di elaborazione del processo. I Job vengono messi in esecuzione a cura di un particolare processo di z/OS chiamato Initiator/Terminator (I/T). L'I/T manda in esecuzione i Job prendendoli dalla Execution Queue. E' importante notare che per il mondo Batch l'AS è creato per una istanza di Initiator/Terminator. L'I/T provvede a caricare nel proprio AS il job batch. Da questo momento il processo attivo nell'AS sarà il Job stesso. Finita l'esecuzione del Job, l'I/T tornerà a prendere il controllo del proprio AS e, se c'è un altro Job nella Execution Queue, provvederà a iniziare di nuovo il processo. Se non c'è nulla in attesa di esecuzione, l'I/T resterà in attesa nel proprio AS di nuovi Job da caricare dalla Execution Queue. I dati vengono elaborati e viene prodotto un risultato. Il risultato di un processo batch non è necessariamente una stampa, esso può essere un nuovo archivio, o una serie di modifiche ad un archivio esistente, il riordinamento di un archivio secondo una chiave di ordinamento (Sort) oppure una combinazione di tutti questi.
- **Output:** La fase di Output pone in un'apposita coda tutti i file di stampa prodotti dall'esecuzione: ad esempio, se il processo prevede la produzione di una stampa essa viene "parcheggiata" su un archivio temporaneo su disco in attesa di successive decisioni. In tale fase può essere visualizzata, effettivamente inviata ad una stampante per una fase di hardcopy o cancellata mediante una fase di **purge** (vedi figura 17).
- **Hardcopy:** E' la fase che produce fisicamente la stampa, cioè invia il file di stampa sulla stampante fisica.
- **Purge:** E' la fase in cui il Job viene cancellato dal sistema JES2.

Tutte le elaborazioni del JES2 si basano su "**code"**, in particolare notiamo code di "input" che contengono i processi in attesa di esecuzione, code di "output" e code di

"stampa". I processi interni del JES2 che fanno passare i Job da una fase alla successiva si chiamano JES2 Processor.

L'insieme delle code del JES, tutto il loro contenuto, le stampe e il JCL in attesa di elaborazione sono contenuti in uno spazio disco predefinito che prende il nome di "**File di Spool"** o semplicemente spool.

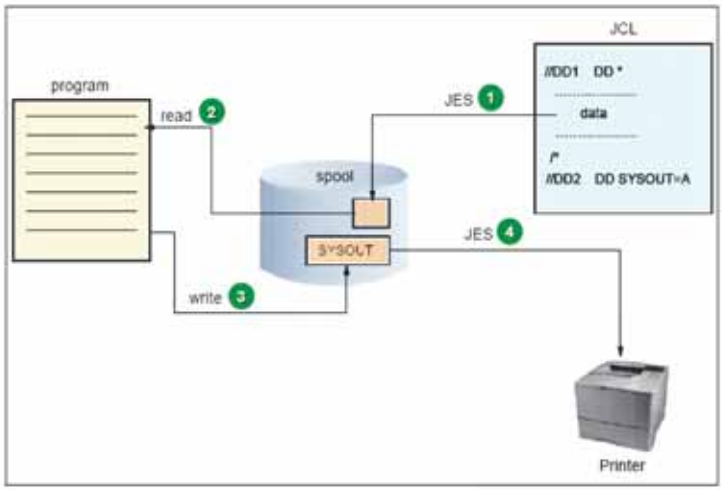


Figura 17 Funzionamento di un programma Batch

3.0.5 Job Control Language (JCL)

Il Job Control Language (JCL) è il linguaggio di controllo interpretato dal JES2; esso deve fornire al JES:

- le indicazioni circa l'ambiente di esecuzione di ogni lavoro batch (JOB)
- la sequenza di esecuzione delle fasi di uno stesso lavoro (Job Steps)
- le indicazioni di controllo per la esecuzione del JOB
- il nome dei programmi da eseguire e le caratteristiche di esecuzione.

Il JCL è un linguaggio procedurale basato su alcune semplici regole con una sintassi estremamente semplice. Gli statement sono record di 80 caratteri; i caratteri permessi sono quelli dell'alfabeto inglese in maiuscolo più caratteri speciali e numeri; è possibile continuare uno statement JCL su più righe, esiste la possibilità di indicare variabili simboliche che vengono istanziate al momento della conversione.

Ogni statement JCL inizia col carattere "//" seguito da una label e da uno dei verbi:

- JOB - Definisce il nome del processo e le caratteristiche di esecuzione
- EXEC - Definisce il nome del programma da eseguire
- DD - Definisce il nome degli archivi e le caratteristiche degli output da produrre. La label di uno statement DD (altrimenti nota come DDName) è decisa da chi ha scritto il programma indicato alla scheda EXEC. Questa

label è di fatto l'aggancio fra il programma ed il particolare file che si vuole usare per questa esecuzione del programma. Una delle voci fondamentali della descrizione di un programma è quali label debbono essere usate per accedere a quale set di informazioni.

La figura 18 illustra questa relazione.

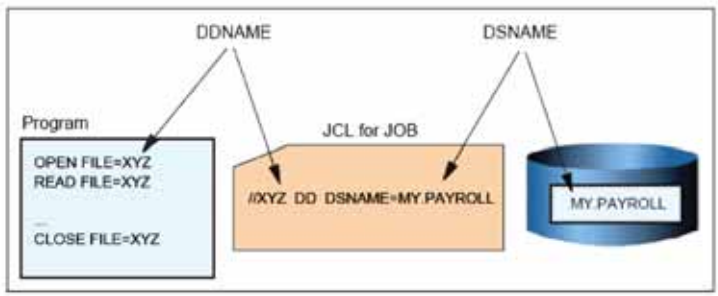


Figura 18 Nomi simbolici e reali in JCL

E' un'importante caratteristica dei Job batch ed in generale di tutti i programmi in esecuzione su z/OS di referenziare al loro interno sempre nomi generici (simbolici) che non hanno riferimento con i nomi degli archivi sui dischi. Ciò consente di usare la stessa copia di programma con archivi diversi in diversi momenti usando diverse JCL.

Analizziamo come esempio il semplice JCL:

```
//MYJOB   JOB 1
//MYSTEP  EXEC PGM=SORT
//SORTIN  DD DISP=SHR,DSN=ZPROF.AREA.CODES
//SORTOUT DD
DISP=SHR,DSN=ZPROF.AREA.NEWCODES
//SYSOUT  DD SYSOUT=*
//SYSIN   DD *
   SORT FIELDS=(1,3,CH,A)
/*
```

Il nome che il JES2 assegnerà al JOB è "MYJOB"; esso è costituito da un unico "passo" o step denominato MYSTEP, scopo dello step è di eseguire un programma di nome sort su un archivio di INPUT che il programma indica con SORTIN e che sui dischi magnetici ha il nome "ZPROF.AREA.CODES". Questo archivio di INPUT viene riordinato secondo le specifiche indicate nel file puntato dalla scheda DD di nome SYSIN. Per semplicità queste specifiche sono incluse nel file JCL stesso; nell'esempio si richiede che il contenuto del File di Input venga riordinato in chiave ascendente considerando i campi nelle prime 3 posizioni. Il File riordinato viene ricopiato sull'archivio già esistente (DISP=SHR) che ha per nome: "ZPROF.AREA.NEWCODES". Tutti gli

eventuali messaggi prodotti dal programma "SORT" vengono inviati al JES che li porrà su una coda di output (SYSOUT=*).

La sequenza di caratteri "/*" indica la fine degli statement che costituiscono il file SYSIN.

Esistono alcuni DDName riservati per usi specifici come i seguenti:

//JOBLIB
//STEPLIB
//JOBCAT
//STEPCAT
//SYSUDUMP
.....

Essi non sono in effetti nomi interni di nessun programma ma indicazioni particolari relative a "cataloghi" o "librerie" dove sono posti programmi e dati riservati a particolari funzioni diagnostiche (Dump).

Un particolare tipo di JCL è quello che costituisce una "procedura" detta PROC. Una PROC è un particolare file JCL che viene memorizzato in maniera permanente in una area di disco denominata Libreria delle procedure (PROCLIB). La procedura può essere richiamata direttamente dalla console di sistema mediante il comando "Start *nome della procedura"*. Un Job eseguito con questa modalità prende il nome di "Started Task". Lo z/OS assegna un AS ad ogni Started Task.
Le procedure PROC vengono usate per eseguire lavori ripetitivi nei quali solo uno o più elementi sono variabili oppure sono usate per avviare funzioni di sistema.

Ad esempio analizziamo la PROC:

```
//MYPROC  PROC
//MYSORT  EXEC PGM=SORT
//SORTIN  DD DISP=SHR,DSN=&SORTDSN
//SORTOUT DD DISP=SHR,DSN=&SORTNEW
//SYSOUT  DD SYSOUT=*
//SYSIN   DD DISP=SHR,DSN=&PARM
//        PEND
```

Questa PROC esegue le stesse funzioni dell'esempio precedente salvo il fatto che essa può essere invocata in modo parametrico attraverso le variabili simboliche:

&SORTDSN che rappresenta il nome dell'archivio di input
&SORTNEW che rappresenta il nome dell'archivio di Output
&PARM che rappresenta un archivio che contiene le direttive di controllo del programma (in particolare la stringa `SORT FIELDS=(1,3,CH,A))`. Supposto che

queste ultime siano contenute in un archivio di nome ZPROF.SORT.PARAMETERS, si ottiene lo stesso effetto dell'esempio precedente con il comando:

```
START MYPROC,SORTDSN=ZPROF.AREA.CODES,
PARM=ZPROF.SORT.PARAMETERS,
SORTNEW=ZPROF.AREA.NEWCODES
```

Una delle funzioni del Converter/Interpreter è quella di risolvere tutti i parametri simbolici all'interno di una PROC.
Spesso la sottomissione di un JCL al JES2 o l'esecuzione di una PROC possono dare lo stesso risultato, come nel nostro esempio, tuttavia ricordiamo che il comando "submit" da TSO/ISPF viene utilizzato prevalentemente per lavori tipicamente "batch" come stampe ed elaborazioni periodiche, manutenzione e copie, mentre il comando START e le PROC vengono usate per i sottosistemi, i programmi di servizio o i middleware che generalmente vengono avviati una sola volta e mai interrotti.

Ad esempio all'avvio del Sistema (IPL) i comandi:

```
START TSO
START VTAM,LIST=AA
START TCPIP
START RMF,PARM=00
```

servono ad avviare rispettivamente il TSO (La funzione TSO/E), il VTAM (gestore del protocollo SNA), il gestore del TCP/IP e uno strumento software di misura del Sistema denominato Resource Monitor Facility (RMF). A ciascuno di essi corrisponde una PROC con i nomi TSO, VTAM, TCPIP, RMF ed alcune di esse hanno la possibilità di inserire parametri come la lista di partenza del VTAM e un file di parametri per RMF.

3.0.6 Spool Display & Search Facility (SDSF)

La componente del Sistema z/OS denominata Spool Display and Search Facility (SDSF) contiene l'interfaccia utente a pannelli o grafica per interfacciarsi con le altre componenti di sistema e con il JES2: analogo interfacciamento può anche avvenire attraverso la System Console o un terminale TSO.

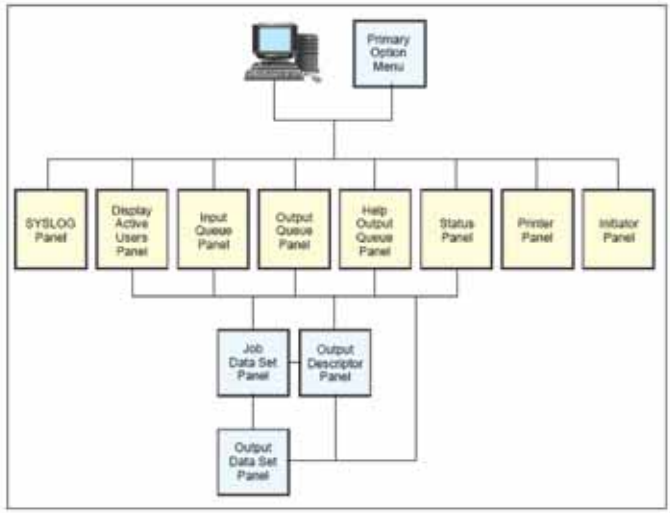


Figura 19 Scelte possibili di SDSF

SDSF è una applicazione eseguita sotto TSO/ISPF e pertanto richiede un terminale di tipo 3270, reale o emulato. Si tratta di una interfaccia a pannelli concatenati che consentono di accedere a tutte le funzioni principali del Sistema e del JES2, fornendo anche supporto per il controllo delle autorizzazioni o delle abilitazioni alle quali l'utente è sottoposto.

La Figura 19 illustra le scelte possibili di SDSF.

Il look del pannello principale di SDSF è riportato nella Figura 20.

Figura 20 Pannello principale SDSF

Tra le funzioni principali di SDSF ricordiamo:

- L'accesso alla System Console (Syslog) in grado di fornire comandi al sistema come se si fosse fisicamente collegati alla Console. In questo caso i comandi debbono essere preceduti dal carattere "/"
- L'accesso allo SPOOL per visualizzare le code dei lavori sottomessi al JES2 ed in attesa di essere elaborati (Comando INPUT)
- L'accesso ai lavori già eseguiti ed alle stampe prodotte (comando OUTPUT)
- L'interrogazione di tutti i lavori attivi (Comando DA)
- L'interrogazione sullo stato di un lavoro (Comando ST)

Dai pannelli SDSF è possibile fornire comandi di linea che consentono di filtrare i contenuti (Comando PREFIX), o di visualizzarli direttamente sul terminale video dopo averli selezionati (Comando SELECT).

Mediante SDSF è anche possibile visualizzare tutti gli address spaces attivi nel Sistema, anche quelli non iniziati dal JES2 e non relativi a lavori Batch o a started task; su di essi è possibile intervenire variandone alcune caratteristiche o cancellarli. Se un address space viene cancellato tutti i lavori in esso eseguiti vengono terminati immediatamente, ciò potrebbe arrecare danni significativi al Sistema e quindi la cancellazione di un address space è un'attività sottoposta a controlli. Alcuni address spaces non sono cancellabili.

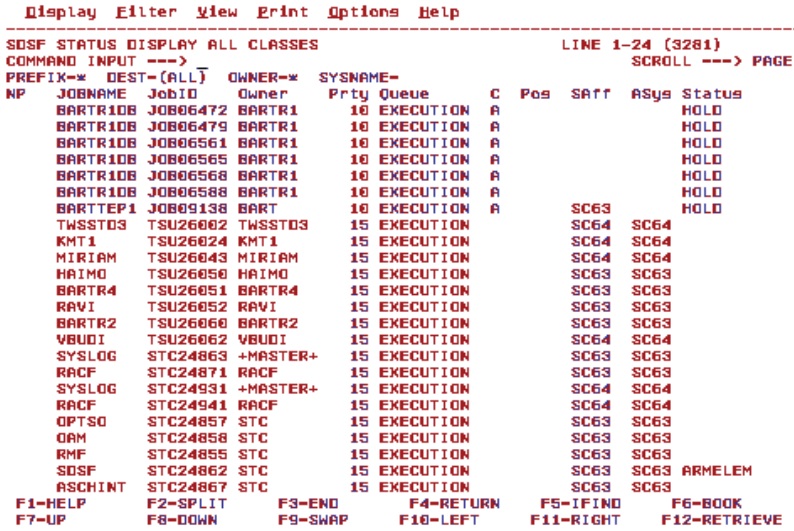


Figura 21 Schermata SDSF - DA

La figura 21 mostra il risultato del comando di visualizzazione di tutti i lavori attivi (DA). Esso consente di visualizzare tutti gli address spaces presenti nel Sistema.

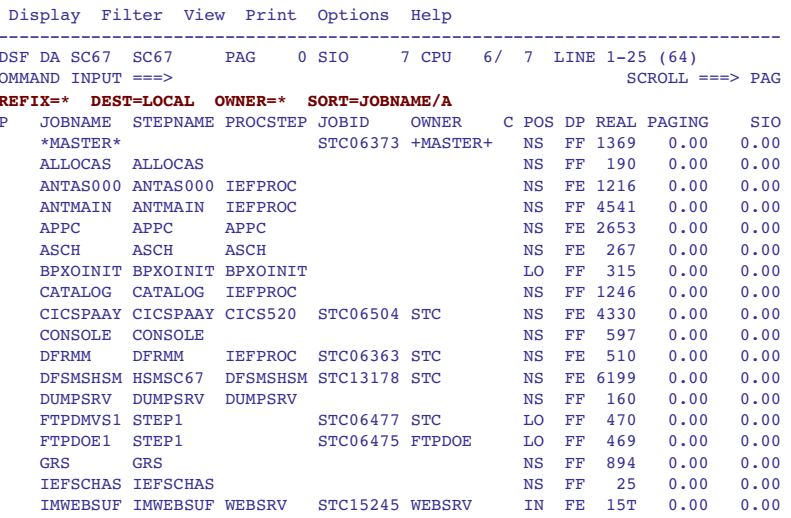


Figura 22 Schermata SDSF

Il Comando "Status", (ST) in Figura 22, consente invece la visualizzazione dello stato dei lavori sottomessi compresi quelli già conclusi; questi ultimi sono visualizzati con un diverso colore. Selezionando una riga si possono vedere tutte le stampe prodotte da un lavoro con vari livelli di dettaglio. La figura 23 mostra la richiesta e la relativa risposta di un comando SDSF.

Screen 2

Figura 23 SYSOUT dataset del job MIRIAM2

3.0.7 Il Workload Manager (WLM)

Il Workload Manager (WLM) è il componente di z/OS responsabile della gestione delle risorse disponibili (CPU, Memoria, I/O) in funzione del raggiungimento di obiettivi di prestazione (Goal).

Il suo compito principale quindi è quello di ottimizzare l'utilizzo delle risorse disponibili in modo da consentire il raggiungimento dei livelli di servizio prefissati per le applicazioni presenti in un sistema.
Il livello di servizio è espresso con una delle due unità di misura:

- Tempo di risposta. Viene usato per le applicazioni transazionali in cui è la durata dell'interazione con l'utente finale può variare solo in funzione dello stato del sistema; è quindi possibile stabilire una soglia da non superare che definisce il livello di servizio accettabile.
- Velocità di esecuzione. Viene usato per i lavori batch la cui durata non è definibile a priori e soprattutto può variare sensibilmente (l'elaorazione trimestrale dei conti correnti dura molto più a lungo dell'elaborazione settimanale degli stessi).

Il Workload Manager opera come segue:

- Controlla costantemente (Monitoring) l'uso delle risorse da parte di ogni Address Space.

- Misura il livello di servizio di ogni applicazione e lo confronta con gli obiettivi stabiliti. In funzione di questa analisi:
    - Determina quali Address Spaces debbono essere rallentati e di quanto.
    - Determina quali Address Spaces debbono essere accelerati e di quanto.
    - Limita la creazione di nuovi address spaces o avvia processi di page stealing nel caso vi sia carenza di memoria
    - Cambia le priorità di esecuzione degli Address Spaces (dispatching priority) in base a quante risorse ciascun address space sta consumando.

Per raggiungere i suoi scopi il WLM scambia informazioni con tutte le altre componenti del Sistema Operativo e dei sottosistemi , ad esempio viene "informato" quando:

- Una parte della memoria viene rimossa dal Sistema
- Un nuovo address space è stato creato
- Un Address Space è stato cancellato
- Una operazione di swap out è stata completata
- Un dispositivo di I/O è stato allocato

Il Workload Manager pensa sempre in termini di Cluster (Parallel SYSPLEX – v. più avanti). Il Workload manager interagisce con i vari gestori di lavori attivi in un sistema z/OS, quali il JES2 già visto e i Transaction managers che vedremo. Infatti il WLM può fornire a questi statistiche su come ogni sistema all'interno del SYSPLEX sta raggiungendo gli obiettivi.
Le politiche su cui si basa il WLM (Policies) sono sempre applicate all'intero SYSPLEX. Un caso particolare può essere un SYSPLEX con una sola Partizione (Monoplex).
Inoltre il WLM interagisce con il PR/SM per riequilibrare i pesi di una LPAR rispetto ad un'altra sulla stessa macchina se questo si rendesse necessario per mantenere gli obiettivi prefissati.

Per decidere le politiche di esecuzione dei lavori è necessario fornire al WLM alcune informazioni e dargli degli obiettivi. Questa modalità operativa si definisce "Goal Mode".

Il WLM ha un'interfaccia molto semplice sotto TSO/ISPF mediante la quale l'utente del Sistema Operativo (System Programmer) può definire le politiche di esecuzione definendo i livelli di servizio (Tempi di risposta, Velocità di esecuzione) e le regole con cui i lavori che entrano nel sistema sono classificati. Sarà poi compito del WLM tradurre le richieste in parametri interni allo z/OS e controllare che essi siano rispettati.

Sotto il controllo del WLM un Cluster di sistemi z/OS utilizzerà al meglio le risorse disponibili e si accorgerà subito di quanto esse siano adeguate o inadeguate agli obiettivi prefissati e richiesti.

La funzione di controllo sull'ottenimento dei risultati è importante anche per stabilire se gli obiettivi prefissati sono consoni alle risorse disponibili, ovvero quali risorse devono essere aumentate.

3.0.8 Gestione dei dati in z/OS

In questo capitolo ci occuperemo sommariamente della gestione dei dati sul Sistema z/OS e delle principali strutture che descrivono la configurazione di un sistema z/OS.

I dati cui il SW accede risiedono su dispositivi ad accesso diretto (Direct Access Storage Device – DASD – come vengono chiamati nel mondo mainframe) o ad accesso sequenziale (nastri magnetici contenuti in cartucce). Oggi come DASD si usano i dischi. Sta emergendo un nuovo tipo di supporto: le memorie flash che però non tratteremo in quanto non ancora strategici. L'organizzazione fisica di questi supporti e alcuni esempi di realizzazioni concrete sono spiegate nell'appendice C 'I sottosistemi Storage'.
I dati contenuti su dischi e nastri sono organizzati in "Volumi". Lo z/OS associa un volume ad un device (disco o nastro). Poichè esiste questa corrispondenza uno a uno device-volume, dal punto di vista dell'architettura, si può dire che ogni volume ha un suo Device Number. Alcune differenze nell'uso/gestione fra dischi e nastri:

- L'associazione fra device disco e volume non cambia mai (fixed media); l'associazione fra device nastro e volume cambia continuamente (removable media). Il volume del device nastro è la cartuccia montata nel meccanismo di trasporto.
- I dati su disco possono essere acceduti in modo casuale, mentre i dati su nastro debbono essere acceduti in modo sequenziale.
- Un volume disco può essere allocato (assegnato in uso) a più processi contemporaneamente mentre un volume nastro può essere allocato ad un solo processo alla volta. Lo z/OS verifica autonomamente che questo requisito sia rispettato.

Ogni volume ha una label di 6 caratteri (max); questa label viene chiamata Volume Serial Number (VOLSER), e fa parte di un blocco (VOLSER appunto) scritto sul volume in una posizione ben definita.
Perchè un volume possa essere usato dallo z/OS deve avere un VOLSER e un particolare file chiamato Volume Table of Content (VTOC). La VTOC è un file in cui sono registrati due tipi di informazioni:

- Tutti i Data Set esistenti sul volume con le loro caratteristiche (dove si trovano sul volume, organizzazione dei dati al loro interno, etc.)
- Lo spazio libero sul volume per poter allocare nuovi data set.

La posizione della VTOC all'interno del volume non è fissa come è invece la posizione del VOLSER; perciò la sua posizione è contenuta nel VOLSER.

I VOLSER acceduti da una singola istanza di z/OS debbono essere univoci; ovvero un'istanza di z/OS non accetta di vedere due volumi con lo stesso VOLSER.
Su ogni volume, sia su disco che su nastro, possono essere contenuti "archivi" o files di natura e lunghezza differenti. I files sotto z/OS si chiamano "Dataset" ed il loro nome DSNAME. Il DSNAME ha un formato molto preciso:

- Non può avere più di 44 caratteri
- E' sempre maiuscolo
- Si possono usare caratteri Alfanumerici e alcuni caratteri speciali
- Il DSNAME è suddiviso in parti lunghe fino a 8 caratteri chiamate Qualifier o Qualificatori
- Queste parti sono separate dal punto "**.**"

- Particolare importanza riveste il Qualifier più a sinistra che viene chiamato High Level Qualifier (HLQ): esso è il fondamento su cui si basa la ricerca del DSNAME nel catalogo (vedi oltre). Ad esempio OPER.STIPENDI.MAGGIO è costituito da tre qualificatori. L'HLQ è OPER.

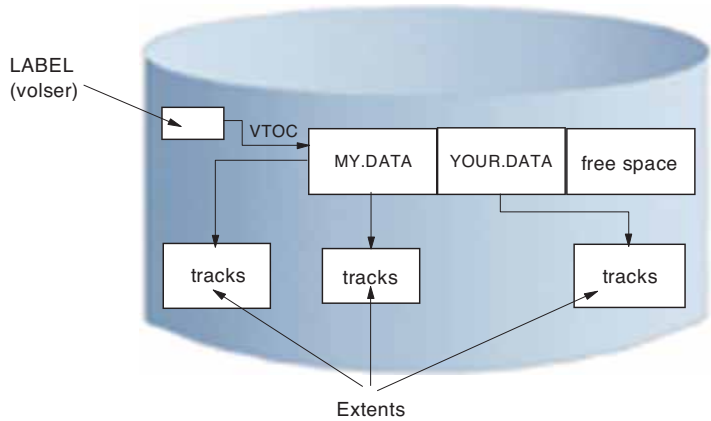


Figura 24 VTOC e Data Set

Dalla figura 24 vediamo che un data set occupa lo spazio su disco in quantità "atomiche" di spazio contiguo chiamate **Extent**. Il numero massimo di extent per data set è limitato a seconda dell'organizzazione del dataset. Un Data Set può avere i suoi Extent residenti su più di un volume (se le dimensioni lo richiedono). Un extent è sempre contenuto in un solo Volume e la sua dimensione è variabile da un Data Set ad un altro. Gli Extent possono non essere contigui. Tutti gli extent di un data set sono descritti in VTOC. Gli extent contengono solo i dati; non contengono nessun metadato che leghi due extent facenti parte dello stesso data set fra di loro. Questi metadati si trovano nella VTOC. Il blocco che li contiene si chiama Data set Control Block (DSCB).
La VTOC registra sia gli extent assegnati ai Data Set sia gli extent non assegnati.
All'interno di una VTOC (e quindi all'interno del volume corrispondente) i DSName non possono essere duplicati.
L'assegnazione di un extent a un data set (anche del primo) è una fase importante del processo di allocazione di un data set. La funzione z/OS che assegna gli extent ai data set è il Direct Access Device Space Manager (DADSM)
Il processo è il seguente:

1. Da parte dell'applicazione viene chiesto al DADSM di assegnare un extent di "x" MB al data set "n".
2. Il DADSM sceglie un volume su cui assegnare l'Extent. Se il Dataset esiste già (cioè è una richiesta per estendere il Dataset durante la fase di scrittura) il volume è quello su cui il data set risiede. Altrimenti (si vuole creare il Dataset) il volume viene scelto secondo criteri definiti dall'installazione.
3. Si analizza la mappa degli extent liberi in VTOC e si individua dove posizionare l'extent richiesto.
4. Viene aggiornata la mappa degli Extent liberi (dello spazio non è più disponibile per nuove assegnazioni)

5. Viene aggiornata la lista degli Extent assegnati al data Set.

Da questa sommaria descrizione si vede come la VTOC sia un elemento fondamentale nella gestione dello spazio su disco.

I cataloghi z/OS

Le dimensioni dei centri mainframe oggi sono tali che c'è bisogno di qualche struttura di metadati che leghi in modo inscindibile il Data Set name al volume (o ai volumi) su cui risiede: questa struttura esiste e prende il nome di sistema di cataloghi.
Questo sistema, (che chiameremo per brevità Catalogo) gestito da un apposito Address Space (Catalog Address Space), costituisce una struttura complessa in grado di reperire velocemente le informazioni sul dataset ed in particolare i suoi extent a partire dal nome, senza esaminare la VTOC di tutti i dischi collegati. VTOC e catalogo insieme permettono al SW di accedere alle informazioni contenute in un Data set solo conoscendone il nome. Inoltre in un sistema di cataloghi i Data set name debbono essere unici (mentre sarebbe in teoria possibile avere Data set name uguali purche' su volumi differenti). Nel Catalogo l'informazione fondamentale è il DSName e il/i volume/volumi su cui risiede. In questo modo la ricerca di un Data Set è:

1. Sul Catalogo
2. Sul volume su cui il Catalogo indica trovarsi il Data Set.

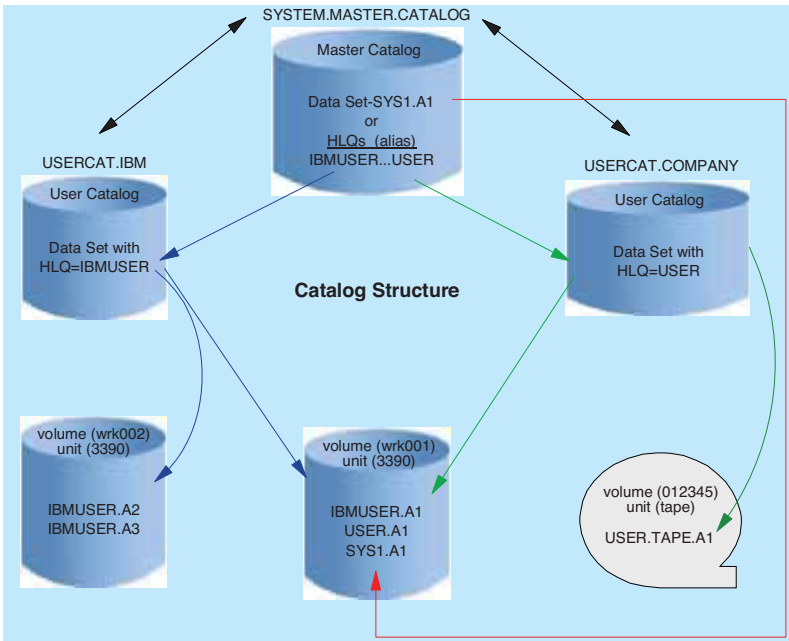


Figura 25 Struttura a cataloghi

Il sistema di cataloghi si basa su una serie di Data Set (Cluster VSAM – vedi oltre) raggruppati su due livelli, come illustrato in Figura 25:

- Master Catalog: E' il punto di partenza per tutte le ricerche. Per ogni z/OS ci può essere un solo Master Catalog. L'indicazione del Master Catalog (DSName e volume su cui si trova) deve essere fornita al sistema in fase di IPL. Se la ricerca del Data set non si conclude sul Master, si procede al secondo livello (se possibile)
- User Catalog: Rappresentano il secondo e ultimo livello. La ricerca non può proseguire a valle di uno User Catalog.

Un Catalogo può essere Master o User secondo cosa l'installazione decide. Però in un Master Catalog debbono essere presenti almeno queste informazioni:

- I Data Set di sistema
- I Connettori agli User catalog che costituiscono il secondo livello di ricerca
- I Data Set di paginazione

Sui cataloghi potranno essere inseriti anche nomi di Data Set su nastro. L'operazione di inserimento di un DSName in un catalogo prende il nome di catalogazione (Catalog), il suo disinserimento prende il nome di scatalogazione (Uncatalog). Eliminare un Data Set da un catalogo non vuole dire cancellarlo dal disco ma semplicemente renderlo "invisibile" alla funzione di ricerca tramite catalogo: si potranno avere quindi files su disco non catalogati (e pertanto accessibili solo attraverso la VTOC) oppure entrate di catalogo che non corrispondono a nessun Data Set. Di fatto se un Data Set è assente dal catalogo è perso visto l'elevatissimo numero di Volumi (migliaia e migliaia) oggi presenti in un sistema z/OS.

L'aggancio fra un Master Catalog e i suoi User catalog avviene mediante i Connettori. Facendo riferimento alla Figura 25 vediamo il funzionamento dei connettori. Prendiamo ad esempio il Data Set IBMUSER.A1. I passi per la ricerca di questo dataset sono:

1. Si cerca nel Master Catalog (SYSTEM.MASTER.CATALOG) se esista un connettore ad uno User Catalog per l'High Level Qualifier IBMUSER
2. Poiché questo connettore esiste e punta allo User Catalog USERCAT.IBM la ricerca proseguirà all'interno di USERCAT.IBM
3. Trovata, in USERCAT.IBM, la descrizione di IBMUSER.A1 si andrà sul volume indicato (WRK001) per verificare se effettivamente il Data Set è lì. Se non c'è, viene riportata la condizione di Data Set not found al processo che aveva chiesto di accedere ad IBMUSER.A1
4. Se il connettore IBMUSER non si fosse trovato, la ricerca per IBMUSER.A1 si sarebbe conclusa nel Master Catalog.

L'insieme delle funzioni di gestione e reperimento dati attraverso cataloghi prende il nome di Integrated Catalog Facility o ICF.

Tipi di Data set z/OS

I Data Set presenti sui dischi gestiti da z/OS possono avere delle strutture differenti in base all'uso che di essi viene fatto. I dati all'interno di un Data Set sono acceduti dalle applicazioni non byte per byte ma aggregati in unità logiche chiamate Record. Tutte le applicazioni accedono alle loro informazioni organizzandole in unità chiamate record logici il cui contenuto e' dettato unicamente dalle esigenza delle applicazioni che vi accedono. Così avremo un record logico *anagrafica* o *conto corrente* o *stipendio.* Questi record logici sono a loro volta aggregati in blocchi prima di essere scritti su disco o nastro. Siccome le dimensioni dei record logici sono sempre piuttosto piccole, se questi venissero scritti singolarmente su disco avremmo un notevole spreco dello spazio di registrazione. Ecco perché i record logici sono organizzati in blocchi fisici. Le possibili aggregazioni Record-Blocchi possono essere (vedi Figura 26):

- FIXED (F): tutti i Record hanno lunghezza eguale. Un blocco contiene un solo Record
- FIXED BLOCKED (FB): tutti i Record hanno lunghezza eguale. Un blocco contiene più di un record. In questi due casi la lunghezza di Blocchi e Record è registrata nel DSCB relativo al Data Set in VTOC
- VARIABLE (V): Ogni Record può avere una lunghezza variabile. Un campo di 4 byte (Record Descriptor Word – RDW) all'inizio del Record indica la lunghezza del Record. Un Blocco contiene un solo record.
- VARIABLE BLOCKED (VB): Ogni Record può avere una lunghezza variabile. Un campo di 4 byte all'inizio del Record indica la lunghezza del Record. Un Blocco contiene più di un record. La lunghezza del blocco è descritta da un campo di 4 byte all'inizio del blocco (Block Descriptor Word – BDW)
- UNDEFINED (U): Non esiste alcuna delimitazione di record all'interno del blocco. La logica per estrarre le informazioni dal blocco fisico è scritta all'interno dell'applicazione che legge e scrive. E' il formato usato per scrivere i programmi eseguibili in una libreria di programmi.
- Nel caso di record variabili o undefined in VTOC viene scritto la massima lunghezza possibile del record e del blocco.

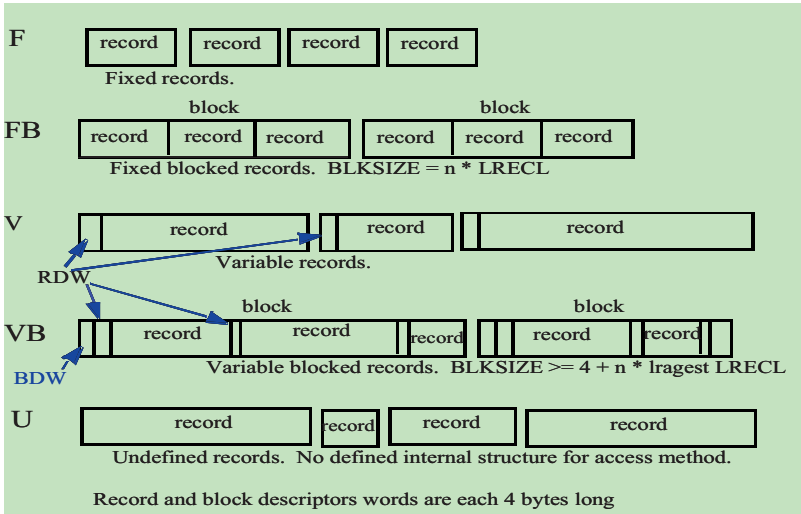


Figura 26 Record e Blocchi all'interno di un Dataset in z/OS

Tra le organizzazioni più comuni di Data Set ricordiamo:

- I **File Sequenziali:** Usualmente F, FB, V, VB, si tratta di semplici raccolte di dati con organizzazione a "RECORD", sono accessibili e visibili direttamente.
- Le **Librerie:** Dette Partitioned Data SET (PDS) o Partitioned data set Extended (PDSE) rappresentano dataset con un ulteriore livello di indice detto "Directory" contenente i nomi di tutti i componenti della libreria detti "membri": essi possono essere programmi eseguibili (LOAD) o programmi sorgenti (SOURCE) o sequenze di JCL (JOB) o procedure (PROC), o files di testo contenenti qualunque informazione o informazioni di controllo o input di altri programmi. I membri delle librerie sono normalmente editabili cioè modificabili attraverso l'editor di ISPF/PDF (questo non si applica per le librerie che contengono programmi eseguibili).
- I Cluster **VSAM:** Virtual Sequential Access Method (VSAM) è un metodo di accesso che consente l'organizzazione dei dati in forma sequenziale con l'introduzione di una o più "chiavi" di accesso dette "indici". Ad esempio il metodo di accesso VSAM consente di archiviare il Nome, l'Indirizzo ed il Numero di Telefono di una serie di persone potendo accedere all'archivio per "NOME", per Indirizzo o per Numero di Telefono senza avviare alcuna operazione di ordinamento dell'archivio stesso. Ogni cluster VSAM deve essere catalogato. Si chiamano cluster poichè sono composti da parti separate dette componenti. Questi componenti possono essere:
    - Data Component (obbligatorio – almeno il data Component deve sempre essere presente) che contiene i dati veri e propri
    - Index Component (facoltativo – costituisce un indice per accedere in modo diretto alle informazioni contenute nel data Component).

Alcuni esempi di Cluster VSAM sono:

- I Data Set di Page. Si tratta di Cluster VSAM di cui esiste solo la parte dati. L'accesso diretto da parte dell'Auxiliary Storage Manager avviene mediante strutture che esistono in Memoria Virtuale.
- I Cataloghi. Sono Cluster VSAM composti dal Data Component (l'insieme delle informazioni sui Data Set) e dall'Index Component per consentire l'accesso diretto alle informazioni. La chiave per ricercare nell'Index Component è il nome stesso del Data set. Un accesso al catalogo richiede nella massima parte dei casi non più di due/tre operazioni di I/O, qualunque sia la dimensione del catalogo stesso.

Si possono copiare dati e membri dalle librerie o agire sui Data Set o sui Files VSAM usando alcuni programmi di Utilità. I più comuni tra questi sono:

- IEBGENER – Copia DataSet Sequenziali
- IEBCOPY – Copia Membri di Libreria
- IDCAMS – Gestisce tutto l'ambiente VSAM, con l'uso di parole chiave come in figura 27.

```
DEFINE CLUSTER -
(NAME(VWX.MYDATA) -
VOLUMES(VSER02) -
RECORDS(1000 500)) -
 DATA -
(NAME(VWX.KSDATA) -
 KEYS(15 0) -
RECORDSIZE(250 250) -
BUFFERSPACE(25000) ) -
INDEX -
(NAME(VWX.KSINDEX) -
CATALOG (UCAT1)
```

Figura 27 Comandi VSAM

Una forma particolare di Cluster VSAM è il Cluster Linear il cui spazio è un area senza alcuna delimitazione in record o blocchi. L'applicazione che usa il Cluster Linear provvede a dare un significato al suo contenuto. Un esempio di questo tipo di Cluster è lo z/OS UNIX File System (zFS).

z/OS UNIX File System (zFS)

Lo z/OS è in grado di ospitare non solo dati "classici" mainframe (scritti secondo l'architettura Mainframe e acceduti da applicazioni Mainframe) ma anche dati UNIX. Questa funzione è necessaria se si vogliono ospitare applicazioni UNIX sotto z/OS.
Uno zFS è un particolare dataset di tipo VSAM Linear in grado di ospitare al suo interno una struttura di File System tipica del mondo UNIX: lo Hierarchical File system (HFS). All'interno di uno zFS viene ospitata la struttura a directory propria del mondo UNIX. Ciascun File system UNIX è contenuto in un Data Set e può essere "Mounted" o "UnMounted" da un Mount Point esattamente come avviene in un mondo UNIX. Esistono utilities che permettono di copiare dati da un file system ad un altro o dall'ambiente UNIX a quello tradizionale e viceversa.

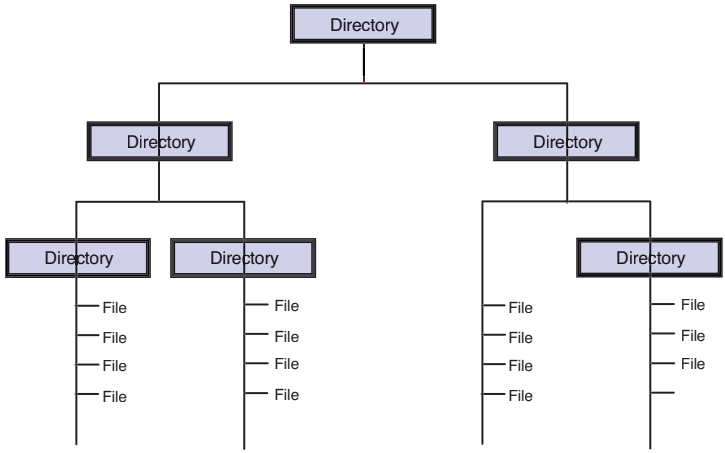


Figura 28 Struttura di un File System

Organizzazione fisica dei dati su disco

Questa è la descrizione di come lo z/OS crede che i dati siano scritti su disco; in funzione di questa tutti i device driver sono costruiti. Vedremo nell'appendice C come questa visione dello z/OS sia realizzata fisicamente all'interno degli Storage Subsystem usando dischi reali di caratteristiche diverse.

Fisicamente un disco è organizzato con 'n' piatti circolari rotanti attorno ad un asse. Ogni piatto offre due superfici magnetizzate per registrare i dati. Su ogni superficie si hanno un numero 'x' di piste di registrazioni circolari (che chiameremo tracce) il cui centro è l'intersezione dell'asse di rotazione con la superfcie stessa.

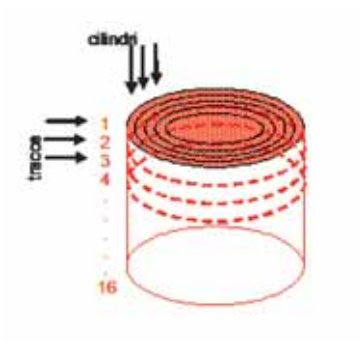


Figura 29 Cilindri e tracce in un disco

Queste tracce hanno una capacità di memorizzazione uguale per tutte, qualunque sia il loro raggio. L'insieme delle piste aventi lo stesso raggio e risiedenti su superfici differenti è il cilindro. La figura 29 illustra questo concetto.

Si definisce geometria di un disco l'insieme di queste tre caratteristiche:

- Numero di byte per traccia
- Numero di tracce per cilindro
- Numero di cilindri per disco

Le informazioni vengono scritte sulle tracce in blocchi. Il blocco è la quantita' (atomica) che viene scambiata fra il disco e il server durante l'esecuzione di un comando di lettura/scrittura. Non è possibile trasferire quantità inferiori al blocco. Questo vincolo dipende dalla natura fisica della registrazione dei dati su disco. La testina di lettura/scrittura ha bisogno di aree con livelli di magnetizzazione differenti (Gap) per delimitare l'inizio e la fine della registrazione di un blocco di dati. Durante la lettura il meccanismo (testina magnetica) ha bisogno di sapere quando comincia il blocco da trasferire e quando finisce: i due gap servono a questo. Siccome lo spazio occupato dai gap non è trascurabile c'è tutto l'interesse ad avere blocchi di registrazione sufficientemente grandi. Ad esempio se il blocco è di 100 byte lo spazio su disco disponibile per le registrazioni si riduce al 30% circa; mentre se il blocco è di 4000 byte lo spazio è utilizzato al 90%.

Per leggere/scrivere un blocco bisogna sapere dov'è/dove scriverlo. Cioè va localizzato. L'unica forma di localizzazione possibile è perciò l'indicazione del cilindro in cui si trova il blocco, della traccia all'interno del cilindro, e della posizione del blocco sulla traccia. Questa localizzazione prende il nome di C-H-R (Cylinder-Head-Record). Ad esempio per il VOLSER:

- C = 0
- H = 0
- R = 3

Cioè il VOLSER è sempre scritto sul cilindro 0, traccia 0, ed è il terzo blocco sulla traccia. C-H-R è l'indicazione di cui ha bisogno l'equipaggio mobile che sposta le testine di lettura/scrittura per posizionarsi correttamente sul blocco da leggere/scrivere. Ma normalmente ci sono altre indicazioni che il SW può usare. Ad esempio, nel mondo FBA, siccome i blocchi hanno tutti la stessa lunghezza è anche possibile puntare al blocco indicando il Block Number dall'inizio del disco (cioè da cilindro "0", Traccia "0" e record "0"). Sarà però necessario per poter arrivare fisicamente al blocco tradurre questa indicazione in C-H-R, poichè questo è l'unico puntamento valido con una struttura sopra descritta.

Architettura Count Key Data

L'espressione Count Key Data (CKD) indica il modo di organizzare la scrittura delle informazioni sui supporti ad accesso diretto nel mondo Mainframe. L'altra modalità di scrittura e' la Fixed Block Architecture (FBA) usata nel mondo UNIX e x86. Ogni blocco di informazioni viene scritto su una traccia nel modo seguente (vedi figura 30).

Figura 30 Il blocco Count Key Data

Abbiamo 3 gap: Prima di Count, fra Count e Key, fra Key e Data.

Il campo Count contiene tre informazioni:

- La localizzazione del blocco nella forma C-H-R
- La lunghezza del campo chiave (se è presente)
- La lunghezza del blocco

Il campo Key (se esiste) può essere lungo fino a 256 byte. Il suo contenuto dipende dall'applicazione che accede al dato. In generale viene usato come argomento per una ricerca del blocco all'interno della traccia.

Il blocco può avere una lunghezza fino a 32KB.

Questa organizzazione permette di avere blocchi di lunghezza variabile, e quindi di sfruttare al meglio lo spazio su disco.

Diciamo subito che il campo Key è una reliquia del passato. Oggi sopravvive in due sole strutture dati su disco:

- Volume Table of Content (VTOC) - Il campo Key contiene il nome del Data Set a cui il blocco si riferisce
- Partitioned Data Set - Il campo Key contiene il nome del membro.

Data set di sistema

Concludiamo la trattazione dell'argomento ricordando alcuni tipi "speciali" di files e librerie presenti nel sistema, in particolare:

- I Files di Configurazione, sono usualmente preceduti dal prefisso SYSx e contengono le informazioni sui devices (SYS1.IODF), sulla configurazione della rete (SYS1.VTAMLIST), o su tutto il Sistema (SYS1.PARMLIB), o le informazioni di avvio dello z/OS (SYS1.IPLPARM)
- Le Librerie di Sistema che contengono le parti principali del Sistema Operativo (SYS1.NUCLEUS) o le procedure principali relative agli "Started task" di Sistema (SYS1.PROCLIB), oltre ai più importanti prodotti software e Middleware, i moduli caricati sempre in memoria all'avvio (SYS1.LPALIB) e alcuni moduli base per la gestione del Sistema (SYS1.LINKLIB).

3.0.9 Gestori di basi dati (DBMS)

Un DBMS (Data Base Management System) è un sistema software progettato allo scopo di garantire la condivisione dei dati, in lettura ed aggiornamento, salvaguardandone l'integrità.

Un database è un insieme integrato di dati, controllato centralmente da un DBMS.

I vari DBMS presenti sul mercato differiscono per le caratteristiche del modello dati supportato.

I primi DBMS, sviluppati a partire dalla metà degli anni '60, supportavano il modello Gerarchico oppure il modello Reticolare. I DBMS oggi più diffusi sono, invece, quelli basati sul Modello Relazionale, sviluppato sulla base di un'idea originale di Ted Codd, un ricercatore IBM, che la pubblicò nel 1969. Dopo un decennio speso nella ricerca e nello sviluppo di prototipi, all'inizio degli anni '80 cominciarono a diffondersi i primi RDBMS (Relational DBMS).

Non tutti i DBMS precedenti, tuttavia, sono stati soppiantati dal nuovo arrivato: alcuni sono sopravvissuti. Nel contesto dell'ambiente z/OS, il DBMS Gerarchico continua ad essere utilizzato da molti clienti soprattutto a supporto di applicazioni che, seppur datate, svolgono ancora una valida funzione di business. Per questo motivo, nel seguito della trattazione, prenderemo in considerazione anche i DBMS Gerarchici oltre a quelli Relazionali.

I Data Base Management System (prodotti da IBM) utilizzati in ambiente z/OS sono:

- DB2 – Data Base Relazionale caratterizzato dal linguaggio di consultazione Structured Query Language (SQL)
- IMS/DB – Data Base Gerarchico caratterizzato dal linguaggio di consultazione Data Language One (DL/1)

Le pagine che seguono descrivono le caratteristiche principali del modello Relazionale e di quello Gerarchico, e dei DBMS appena menzionati.

I DBMS Relazionali

I DBMS Relazionali sono largamente utilizzati in aziende di ogni tipo, come banche, assicurazioni, aziende di trasporto, di telecomunicazioni, ecc. per elaborazioni di tipo transazionali (ambienti OnLine Transaction Processing – OLTP), dove più utenti ovvero più processi richiedono un accesso rapido e, soprattutto, integro a dati condivisi sia in lettura che aggiornamento.

Gli stessi RDBMS usate per l'OLTP sono spesso utilizzati anche per soluzioni di Data Warehousing e Business Intelligence, grazie alla loro capacità di trarre vantaggio, in modo trasparente per l'utente, dalla potenza di calcolo e il parallelismo delle macchine multi-processori e in cluster, indispensabile per rispondere in tempi più brevi o molto più brevi ad interrogazioni che richiedono l'elaborazione di grandi moli di dati.

Il Modello Relazionale è caratterizzato da

1. una struttura formata da tabelle (supportata da definizioni molto rigorose)
2. un insieme di regole atte a salvaguardare l'integrità dei dati a vari livelli (integrità di dominio, di tabella, di correlazione, ecc.)

3. un insieme di operatori per l'accesso e l'aggiornamento dei dati, basati sulla teoria degli insiemi.

CUST_NBR	CUST_NAME	CUST_REGION	CUST_CREDIT
239	TOTALLY WIRED	NE	A
376	BRIKS	NW	AAA
921	BALLOONS, ETC.	SW	AAA
631	GREAT WORKS	SW	A
227	GRATE DANES	SE	AA
338	TODAY'S WORLD	NW	AA

Figura 31 Esempio di Tabella di database relazionale

Uno dei punti cardini dell'idea di Ted Codd, che sta alla base del Modello Relazionale, è l'**Information Principle**[1] secondo il quale "The entire information content of a relational database is represented in one and only one way: namely, as attribute values within tuples within relations".

Diversi sono i vantaggi del Modello Relazionale dei dati rispetto ai Modelli precedenti. Tra i più significativi, che hanno contribuito maggiormente al suo successo, vale la pena di ricordare la facilità d'uso, la flessibilità al cambiamento e, soprattutto, il supporto della cosiddetta "data independence"; ovvero, la separazione molto più netta che nel passato tra il modello logico (la Vista Utente) e il modello fisico (il modo in cui i dati vengono memorizzati) che garantisce la possibilità di variare una serie di elementi di entrambi i modelli (come ad esempio, la possibilità di aggiungere nuove entità o nuovi attributi ad entità già esistenti, a supporto di nuove esigenze di business, o quella di aggiungere, modificare e togliere indici per migliorare, ad esempio, le prestazioni) senza incidere sulla funzione applicativa e quindi senza bisogno di modificare le applicazioni esistenti.

Un ruolo importante e, per molti versi, rivoluzionario rispetto ai sistemi pre-esistenti è stata l'elaborazione per insiemi (e non più una riga alla volta), caratteristica intrinseca del Modello Relazionale, fatta propria anche dallo Structured Query Language (SQL), il nuovo linguaggio che i sistemi Relazionali hanno messo a disposizione per la consultazione e l'aggiornamento dei dati. È proprio questa capacità intrinseca di elaborare per insiemi che ha reso possibile l'elaborazione parallela anche di una singola query, sfruttando in modo trasparente all'utente, tutti i processori disponibili per garantire un tempo di risposta ottimale.

Il linguaggio SQL, originariamente sviluppato nei laboratori IBM, è oggi uno standard di fatto e di diritto.

[1] Si veda, ad esempio, http://www.sigmod.org/codd-tribute.html

I DBMS Relazionali, originariamente nati per la gestione di dati strutturati, oggi sono usati anche per gestire tipi di dati semistrutturati (come, ad esempio, documenti) e non strutturati (come immagini, filmati, ecc.).

Recentemente, gli RDBMS sono stati estesi in modo da supportare anche il formato XML, diventato popolare soprattutto per lo scambio di informazioni in rete, per la maggiore flessibilità di gestione di dati di struttura molto variabile e per il già citato supporto di dati semi-strutturati.

Il Linguaggio SQL

Il linguaggio SQL, uno standard universale per i DBMS Relazionali, è composto di tre componenti, che servono ai seguenti scopi specifici:

- la definizione delle strutture dei dati (**Data Definition Language - DDL**)
- la "manipolazione" dei dati (**Data Manipulation Language - DML**), ovvero, il reperimento, l'inserimento, la cancellazione e l'aggiornamento dei dati.
- il controllo degli accessi ai dati tramite concessione e revoca di privilegi (**Data Control Language - DCL**).

Il linguaggio offre istruzioni specifiche per ciascuna delle tre categorie appena menzionate (vedi figura 32):

- SELECT, UPDATE, INSERT, DELETE per il DML
- CREATE, ALTER, DROP per il DDL
- GRANT, REVOKE per il DCL.

In particolare, le caratteristiche della componente DML permettono di individuare le informazioni da consultare e/o aggiornare senza la necessità di conoscere l'organizzazione fisica del database che le contiene. La scelta della strategia di accesso ai dati per ogni richiesta è a carico dell'Ottimizzatore, una componente peculiare del DBMS Relazionale.

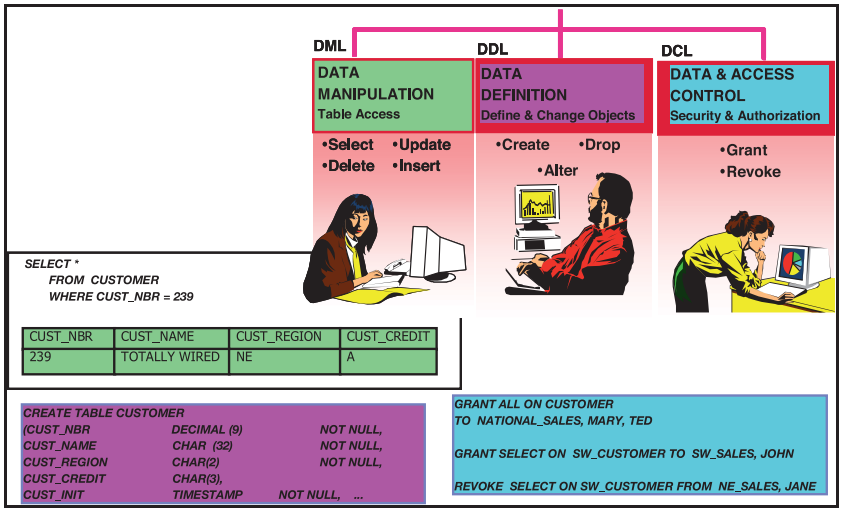


Figura 32 Le componenti del Linguaggio SQL

Le Viste, un altro concetto innovativo introdotto dal Modello relazionale e supportato dal linguaggio SQL, sono lo strumento che permette di raggiungere un livello di Data Independence ancora più elevato rispetto a quello fornito nativamente dall'uso delle Tabelle.

Le Viste possono essere considerate come una finestra logica sui dati: usando gli stessi operatori utilizzati per interrogare la base dei dati, le Viste permettono di selezionare un sottoinsieme di dati (righe / colonne) di una tabella o di più tabelle tra loro variamente e dinamicamente correlate.

Per l'utente, le Viste sono Tabelle a tutti gli effetti[2] grazie ad un'altra caratteristica del Modello relazionale, nota come "Relational Closure", per la quale il risultato dell'uso degli operatori del linguaggio è sempre e comunque una struttura relazionale valida. Questa caratteristica permette un uso *ricorsivo* del linguaggio, che ne aumenta in modo notevolissimo la potenza espressiva.

Nel tempo, il linguaggio SQL standard ISO/ANSI si è arricchito di molte nuove funzioni, tra le quali merita di essere menzionata l'estensione procedurale (Procedural Language – PL) per lo sviluppo di Stored Procedure (SQL/PL), e quella per l'accesso a documenti in formato XML (SQL/XML).

Le Stored Procedure altro non sono che degli eseguibili, sviluppabili appunto in SQL/PL, ma anche in Java oppure in linguaggi tradizionali (Assembler, Cobol, C Language, ecc.). Esse possono essere richiamate tramite una estensione standard del linguaggio SQL da altri programmi, locali o remoti, eseguiti in qualunque ambiente supportato dal DBMS (es. IMS, CICS, TSO, batch) e su qualunque piattaforma supportata.

[2] A parte alcuni limiti al loro uso per aggiornare i dati. Limiti più o meno severi a seconda dell'implementazione

L'uso appropriato delle Stored Procedure permette un più elevato riutilizzo di routine che svolgono funzioni utili condivise da più applicazioni, e un minor traffico di rete rispetto all'esecuzione remota di comandi SQL (vedi figura 33).

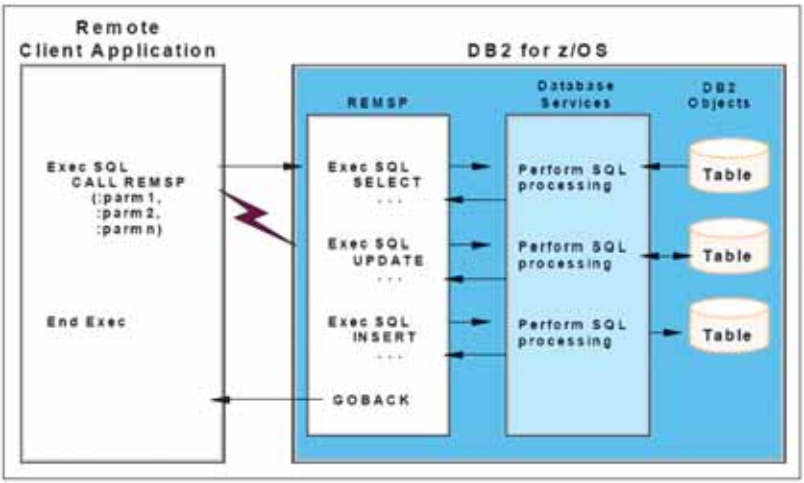


Figura 33 Stored Procedure e Traffico di Rete

IBM DB2 for z/OS

Il DB2 (IBM Database 2), la cui versione per la piattaforma mainframe fu annunciata nel Giugno 1983, è il più importante esponente di una famiglia che opera su tutte le più diffuse piattaforme informatiche.

Il prodotto contiene al suo interno tutte le funzioni più evolute ed è corredato da una suite di strumenti che ne migliorano la gestione e il controllo.

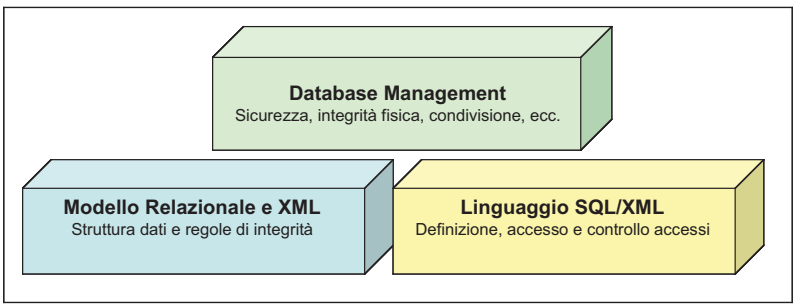


Figura 34 DB2 for z/OS: un RDBMS Ibrido (Relazionale / XML)

Il DB2 for z/OS è dotato di componenti specifiche (Attachement Facilities) per la connessione da parte di applicazioni che operano in ambiente quali CICS, IMS, TSO e batch.

L'integrità della transazione (e, di conseguenza, l'integrità dei dati), è garantita anche nel caso in cui la transazione effettui aggiornamento a più basi di dati locali o remote, grazie al supporto del **two-phase commit**.

La funzione Data Sharing, basata sull'architettura Parallel Sysplex, consente la condivisione integra di un'unica copia dei dati e del relativo Catalogo del metadati da parte di più sottosistemi DB2 che fanno parte di uno stesso Data Sharing Group e operano sulla stessa copia di sistema operativo o su sistemi z/OS distinti, garantendo elevata scalabilità e una continuità di servizio difficilmente eguagliabile su altre piattaforme.

Il DB2 supporta l'accesso a basi di dati distribuite secondo le specifiche della Distributed Relational Data Base Architecture (DRDA). Questa architettura definisce un insieme di protocolli per l'interoperabilità di tipo "client" / "server" tra gestori di dati e client di fornitori diversi. Il DB2 può svolgere sia il ruolo di server nei confronti di client DRDA locali o remoti che il ruolo di client verso altri server DRDA, che operano sia sulla piattaforma z/OS che su altre piattaforme (come AIX, Sun Solaris, HP-UX, Linux, z/Linux, Windows Server).

Il DB2 for z/OS è un sottosistema costituito da diversi componenti attivi in più regioni (Address Spaces) – vedi figura 35:

- **System Services Address Space** (SSAS – procedura xxxxMSTR) che si fa carico delle attività di sistema, come il logging, il controllo delle prestazioni, la gestione dei comandi operativi, ecc.
- **Database Services Address Space** (DSAS – procedura xxxxDBM1) che si occupa più propriamente della gestione dei dati a cura delle funzioni RDS (Relational Data System), Data Manager (DM) e Buffer Manager (BM).
- **Distributed Data Facility Address Space** (DDF – procedura xxxxDIST), che integra sia le funzioni di Client che quelle di Server DRDA.
- **Internal Resource Lock Manager** (IRLM – procedura IRLMPROC) per la serializzazione degli accessi alle risorse condivise.
- **WLM-managed stored procedure address spaces**, utilizzati in numero variabile in funzione del carico di lavoro, per l'esecuzione delle "Stored Procedures"

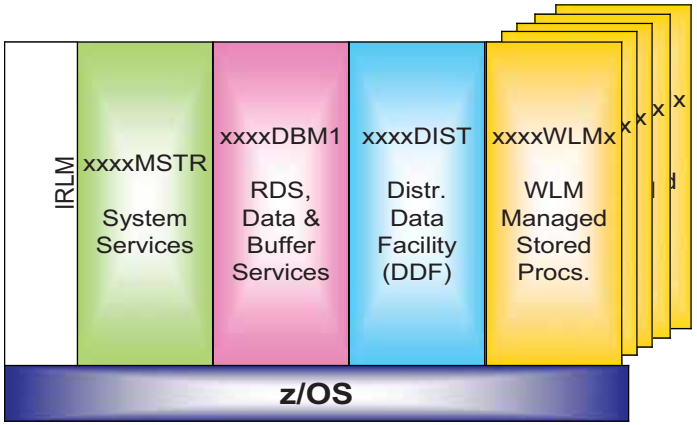


Figura 35 DB2 Address Space

Le tabelle, nelle quali sono memorizzati i dati, si appoggiano su uno o più spazi fisici di tipo VSAM Linear Data Set (LDS) denominati Tablespace.

Gli indici, strutture di tipo B-tree il cui scopo principale è quello di garantire un accesso più veloce ai dati di interesse, sono memorizzati in strutture fisiche separate, costituite sempre da VSAM Linear Dataset.

L'entità Database, invece, a differenza di altri ambienti relazionali, nell'ambiente DB2 for z/OS rappresenta semplicemente un insieme di Tabelle, Indici e Tablespace senza nessun ulteriore estensione a livello fisico (a parte alcune informazioni e blocchi di controllo nelle strutture interne e nel Catalogo dei metadati), con valenza operativa (ad esempio, è possibile attivare e disattivare un Database estendendo automaticamente l'effetto a tutti gli oggetti fisici associati).

Recentemente, il DB2 si è arricchito della capacità di gestire dati in formato XML, permettendone l'accesso tramite il supporto dell'estensione già citata del linguaggio SQL (SQL/XML).

Il DBMS Gerarchico

Il database gerarchico ha caratteristiche fondamentalmente diverse da quello relazionale. Infatti, si adatta a strutture logiche prevalentemente statiche e presuppone che l'utente conosca e navighi nella struttura per raggiungere il record desiderato.

In un database gerarchico le informazioni sono strutturate come in un albero a più livelli, ciascuno caratterizzato dalla presenza di entità dette segmenti. Il segmento al livello più alto della gerarchia prende il nome di radice (o root). Sia la radice che i segmenti dipendenti sono usualmente suddivisi in campi.

La struttura rappresenta intrinsecamente delle relazioni di tipo padre-figlio (ovvero, uno-a-molti, come ad esempio la relazione tra Ordine e Righe dell'Ordine e quella tra

Reparti e Impiegati), nelle quali un padre può avere zero, uno o più figli, ma ogni figlio ha un solo padre.

Ogni albero è formato da un unico segmento radice (detto anche segmento padre o genitore) e da un insieme di segmenti (uno o più) dipendenti da esso, in modo ripetitivo (ogni segmento a sua volta può essere padre di altri segmenti, e così via).

Ciascuna struttura ad albero costituisce, quindi, un insieme organizzato di segmenti o rami strutturati del database. La figura 36 illustra questo concetto.

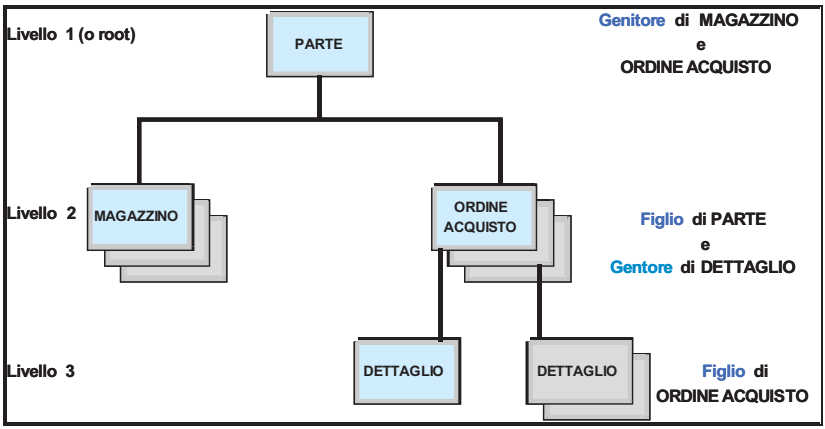


Figura 36 Esempio di database gerarchico

Il Modello Gerarchico si presta bene, ovviamente, alla modellazione di porzioni del mondo di interesse che si presentano, intrinsecamente, come un insieme gerarchico; ma presenta notevoli difficoltà quando l'universo di interesse presenta anche relazioni di tipo molti-a-molti (come ad esempio nel caso, ad esempio, nella relazione Clienti – Ordini – Righe dell'Ordine – Prodotti).

Information Management System/DB (IMS/DB)

L' Information Management System (IMS) è stato disegnato dall'IBM, insieme a Rockwell e Caterpillar, a partire dal 1966, a supporto del programma Apollo.

Nato come sistema esclusivamente batch, arricchitosi successivamente delle componenti OLTP, è ancora in uso in molte delle installazione basate su piattaforma z/OS.

Le prestazioni sono senz'altro uno dei valori specifici di questo sottosistema, che enfatizza ulteriormente questa caratteristica con la funzione "Fast Path", che utilizza strutture dati semplificate e database mantenuti costantemente in memoria per garantire prestazioni ancora più elevate.

L'accesso ai dati avviene tramite un linguaggio proprietario, il Data Language/One (DL/1), che consente di inserire, cancellare, modificare e interrogare le informazioni del database.

L'IMS supporta tre tipi di database a struttura gerarchica:

"Full Function" DB	database con indici primari e/o secondari, supporto di relazioni logiche tra database diversi.
"Fast Path" DB	questa categoria di database non dispone di funzioni di indicizzazione. L'accesso avviene tramite uso di tecniche di randomizzazione della chiave di ricerca. Supporta database costantemente mantenuti in memoria.
High Availability Large Databases (HALDB) - sono una estensione della categoria "Full Function" cui si aggiungono funzioni di alta affidabilità e migliore gestione di grandi volumi di dati.	

3.0.10 Disegnare Applicazioni per z/OS

Realizzare un'applicazione implica la soluzione di un insieme di problemi legati alla complessità dello sviluppo e della gestione del progetto. Sappiamo infatti che una applicazione è costituita da uno o più programmi in grado di soddisfare una specifica richiesta o di risolvere uno specifico problema. Tale soluzione può richiedere risorse di vario tipo allocate su sistemi e piattaforme diverse.

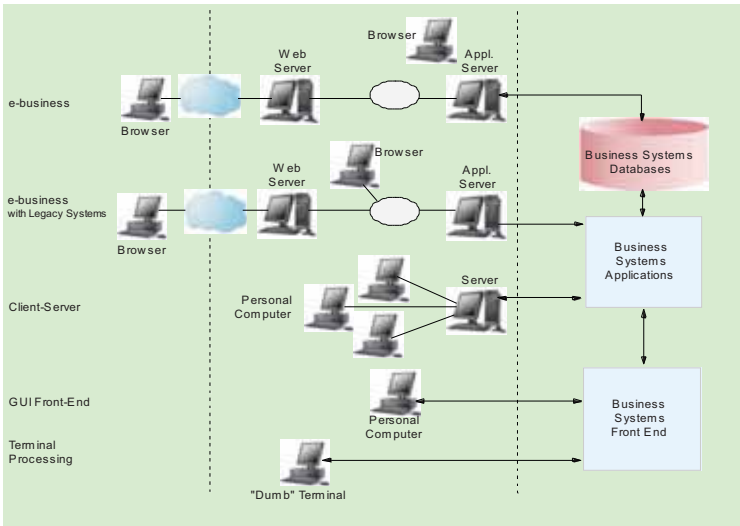


Figura 37 Infrastruttura delle applicazioni

La realizzazione di un progetto informatico di sviluppo applicativo è quindi spesso una operazione che presenta un elevato grado di complessità sia dal punto di vista tecnico che da quello organizzativo. Considerando solo questo ultimo aspetto possiamo individuare due figure professionali:

- **Application Designer -** il cui compito è quello di operare scelte al fine di raggiungere la soluzione ottimale, ovvero, a partire dai requisiti dell'applicazione e da una visione globale del sistema, selezionare la piattaforma e le risorse hardware e software necessarie alla realizzazione.

- **Application Programmer -** il cui ambito di competenza è quello di costruire, provare e rilasciare l'applicazione (o parte di essa) in base alle specifiche del disegno applicativo, utilizzando tutti i tools a disposizione.

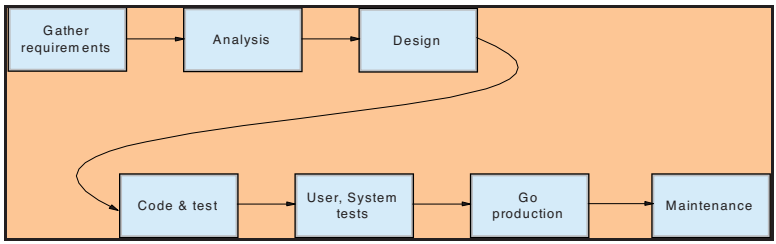


Figura 38 Fasi dello sviluppo di un applicazione

Sempre dal punto di vista organizzativo la programmazione applicativa è costituita da una serie di iterazioni in cui l'applicazione viene suddivisa in componenti omogenee. Quindi a partire dai requisiti richiesti si percorrono le fasi di disegno ad alto e basso livello, scrittura del codice e compilazione delle componenti e loro test. Quindi le componenti vengono integrate e il sistema viene testato in maniera globale. Seguono le fasi di rilascio in produzione.

Per ciò che concerne invece i requisiti di una applicazione, essi in estrema sintesi saranno di tipo Funzionale e Non Funzionale.

I **requisiti funzionali** provengono da una esigenza ben definita di business come ad esempio la gestione di un conto corrente di una banca. Essi descrivono un problema o una funzione ben definita legata alle esigenze aziendali che l'applicazione deve soddisfare.

I **requisiti non funzionali,** egualmente importanti, sono quelli che individuano le caratteristiche di ambiente in cui l'applicazione deve poi operare. Tra queste identifichiamo: affidabilità, interoperabilità, performance, usabilità, portabilità, connettività, livelli di servizio ecc.

Le scelte che in genere L'Application Designer è chiamato a fare sono legate all'ambiente infrastrutturale e alle modalità di esecuzione dell'applicazione. Esempi tipici di alternative da valutare sono legate al tipo di processo (batch o online), tipo di

infrastruttura dati (database, tape, flat file, ecc.), tipo di compilatore (COBOL, PL/I, Java, Assembler, ecc.), sistema operativo (z/OS, UNIX, Linux, Windows), potenza e caratteristiche dei server, utilizzo anche parziale di soluzioni a pacchetto, ecc.

I tool di programmazione sono molteplici e legati alle diverse necessità: esistono editor (ad esempio TSO/ISPF), vari gestori del codice sorgente che tra l'altro consentono di mantenere versioni diverse dello stesso codice (ad esempio il SW Configuration Library manager di IBM – SCLM – e altri), strumenti di debugging e strumenti di monitoring software per controllare l'esito delle attività nelle varie fasi dello sviluppo.

3.0.11 Linguaggi di Programmazione

Il Sistema Operativo z/OS è in grado di supportare diversi linguaggi di programmazione.

- ***First and Second generation*** come:
  - Linguaggio **Assembler** – raramente usato per lo sviluppo di applicazioni. Le sue istruzioni sono:
    - Le istruzioni dell'instruction set z
    - API di interfaccia con z/OS

    Esso viene usato per:

    - accedere direttamente a bit o byte;
    - accedere ai system control blocks;
    - avere tempi di risposta molto ridotti o efficienza elevata;
    - realizzare pezzi di codice riusabili richiamati all'interno di programmi scritti in linguaggi ad alto livello.

- ***Third generation*** (High Level Languages) come:
  - **COBOL** - utilizzato per applicazioni business-oriented; è stato integrato nel sistema z/OS con la capacità di interagire con codice Java, gestire documenti XLM e formati Unicode;
  - **PL/I** ("Programming Language/One) - è un linguaggio disegnato per applicazioni di business o scientifiche. Esso è uno dei più ricchi di funzioni. Supporta la recursività e la programmazione strutturata. Una sua variante chiamata **PL/S**, non commercializzata, viene usata per la scrittura del sistema operativo z/OS;
  - **C language -** usato per applicazioni Scientifiche o Gestionali;
  - **Fortran -** utilizzato prevalentemente per Applicazioni Tecnico-Scientifiche.

- ***Fourth Generation*** (4GL) di natura non procedurale - sono linguaggi che realizzano il paradigma dell'Object Oriented programming; come:

    - **C++** derivato dal linguaggio C;

    - **Java** - caratterizzato da un'alta portabilità tra differenti piattaforme informatiche. Tale portabilità è realizzata attraverso uno strato interpretativo di Software denominato Java Virtual Machine (JVM) il quale realizza il disaccoppiamento tra l'architettura ed il linguaggio. Il Sistema Operativo z/OS è dotato di una JVM integrata ed appositamente ottimizzata. Sotto z/OS Java può essere eseguito direttamente o attraverso appositi Middleware come ad esempio Websphere Application Server (vedi oltre).

Particolare importanza ai fini dell'Automazione del Sistema assumono particolari **Linguaggi Procedurali** di Servizio - si tratta di linguaggi orientati a raggruppare comandi di sistema ed altre funzioni in procedure; tra questi troviamo ad esempio:

- **CLIST** Command List - usato come linguaggio di procedure di comandi per la programmazione delle routine di accesso al TSO/E e per alcune applicazioni ISPF come la creazione di Menu per l'utente o il controllo del flusso applicativo dello stesso;
- **REXX** - è un linguaggio che può essere compilato o interpretato, con maggiori funzioni rispetto al linguaggio CLIST; anch'esso è usato per la programmazione di routine di accesso del TSO/E e per alcune applicazioni ISPF.

Ricordiamo infine un componente del Sistema z/OS denominato **Language Environment**: si tratta della realizzazione di un ambiente di esecuzione comune a tutti i linguaggi di programmazione; esso in particolare combina tutti i servizi essenziali per l'esecuzione dei programmi e la gestione degli errori oltre la gestione della memoria. Crea interfacce comuni consentendo tra l'altro il richiamo di programmi scritti in linguaggi diversi da quello del chiamante.

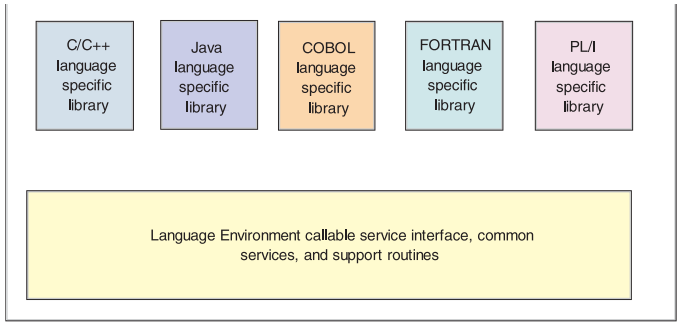


Figura 39 Language Environment

I processi di compilazione in un sistema z/OS di un codice sorgente scritto in uno dei linguaggi appena descritti vengono suddivisi nelle fasi di figura 40.

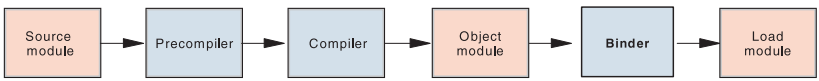


Figura 40 Fasi di compilazione e linkage editor

Il Source Module è un file contenente statement scritti nel linguaggio scelto. Esso è creato dal programmatore applicativo. Il Source Module viene compilato da un Precompilatore (se ad esempio all'interno del Source si usano statement SQL) e dal compilatore vero e proprio che produce un Object Module. Questo è la traduzione in linguaggio macchina degli statement contenuti nel Source Module. Infine l'Object Module viene elaborato dal Binder (o Linkage editor come era chiamato qualche anno fa) che produce il Load Module. Il Load Module viene infine scritto in una libreria di programmi eseguibili da cui sarà poi caricato in memoria virtuale per l'esecuzione quando un processo lo richieda (ad esempio per eseguire un comando TSO/E).

3.0.12 Sottosistemi Transazionali

I sottosistemi transazionali sono particolari programmi che permettono l'esecuzione simultanea di **transazioni** da parte di **molti utenti collegati** al sistema attraverso terminali detti ***workstations***. In effetti non è strettamente richiesto che una workstation sia un terminale video con tastiera, esso potrebbe essere anche un processo automatico, un altro programma eseguito su una differente piattaforma informatica o un dispositivo speciale (ad esempio un ATM, un lettore di tessera magnetica, un rivelatore di RFID, una macchina contapezzi, un telefono cellulare ,ecc.).

Una transazione è un'unità di lavoro indivisibile che comprende più operazioni. Ad esempio una transazione è l'insieme di operazioni che vengono svolte nell'atto di prelevare una somma di denaro contante da un dispositivo "cash dispenser" (ATM); esse sono ad esempio la verifica della disponibilità sul conto corrente, la verifica della disponibilità delle banconote nell'ATM, la verifica del PIN della carta magnetica, l'erogazione del denaro, l'addebito del conto, la registrazione della operazione contabile sulla contabilità dell'agenzia, l'aggiornamento della carta magnetica, ecc.

Le proprietà della transazione sono:

- **A**tomicità, ovvero l'indivisibilità dell'operazione - o è totalmente completata o non è effettuata.
- **C**onsistenza, ovvero l'esecuzione della transazione fa passare il data base da uno stato consistente iniziale ad un diverso stato consistente finale. I criteri della consistenza sono definiti dal Data Base stesso.
- **I**solamento, ovvero ogni transazione non deve interessare altre transazioni concorrenti né esserne influenzata.
- **D**urevolezza, ovvero una volta che la transazione è completata con successo i cambiamenti di stato effettuati (per esempio sul data base, ma non solo) sono duraturi e persistenti.

Il Sottosistema Transazionale controlla la sequenza di eventi che fanno parte di una transazione e si assicura che le caratteristiche della transazione siano mantenute (ACID).

Esso include le funzioni di interfacciamento ai database e il coordinamento del ***commit/rollback*** delle transazioni: cioè è in grado di stabilire quando la transazione è stata completata definitivamente o di "ritornare indietro" nel caso ciò non sia avvenuto per un qualsiasi motivo.

Esso fornisce inoltre una interfaccia comune per i programmi applicativi in modo che le applicazioni possano usare al meglio i servizi di transaction monitor indipendentemente dal linguaggio in cui sono state scritte.

Ciascun Sottosistema Transazionale può supportare un certo numero di linguaggi di programmazione e l'interfaccia con alcuni Gestori di Dati.

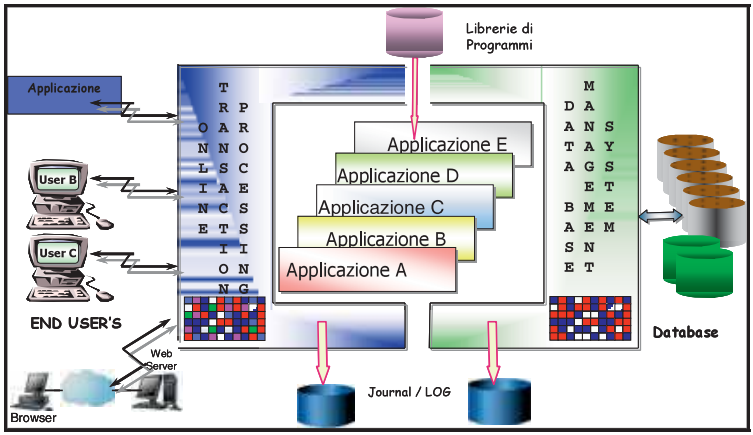


Figura 41 Sottosistemi Transazionali

I principali Sottosistemi Transazionali in z/OS sono:

- CICS/TM - Customer Information Control System/Transaction Manager
- IMS/TM - Information Management System/Transaction Manager
- WebSphere Application Server

CICS/TM Customer Information Control System/Transaction Manager

CICS è un sottosistema di gestione dell'elaborazione delle transazioni online (Online Transaction Processing - OLTP) che fornisce funzioni di middleware per la gestione del sistema e l'accesso ai database.

CICS è utilizzabile da programmi scritti in Cobol, PL/1,C, Java e supporta dati VSAM, Data Base gerarchici (DL/I) e relazionali (DB/2).

Il CICS è un software che si posiziona tra sistema operativo e applicazione. Il programmatore scrive un'applicazione utilizzando le API fornite dal CICS che realizzano le interazioni necessarie con il sistema operativo. Ad esempio, le schermate relative a transazioni in ambito CICS sono inviate e ricevute tramite un emulatore 3270 in forma di mappe. Quando un programma prevede un input o un output di una transazione esso utilizza le mappe richiamandole tramite chiamate al CICS (Command level EXEC).

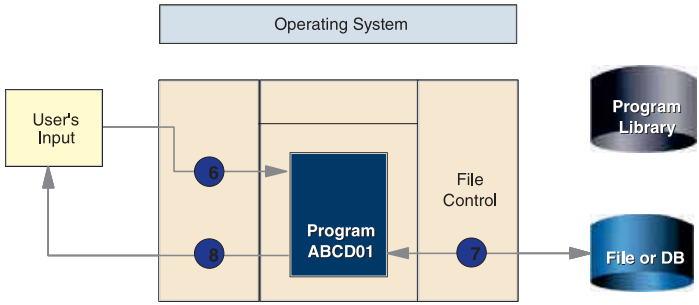


Figura 42 CICS

Una copia di CICS in uno z/OS può gestire le transazioni di molti utenti e molti programmi applicativi. Inoltre più copie di CICS in vari serventi possono comunicare tra loro e condividere dati.

Il CICS supporta il Parallel Sysplex utilizzando il datasharing in sinergia con DB2, IMS/DB e i VSAM (Virtual Storage Acces Method) data set.

Sotto la stessa istanza di Sistema Operativo z/OS possono essere attivate molte istanze di CICS (dette anche regioni), ciascuna istanza risiede in un suo Address Space.

Grazie a questa caratteristica, all'interno di un sistema z/OS spesso viene utilizzata una configurazione con regioni CICS multiple. Queste regioni gestiscono uno stesso ambiente transazionale ma sono specializzate secondo le tre funzioni:

- Terminal Owning Region (TOR) - gestione delle comunicazioni e dei terminali
- Application Owning Region (AOR) - gestione delle transazioni e dei programmi
- Data Owning Region (DOR) - gestione dei dati

In più il CICS fornisce il framework per lo sviluppo, messa in produzione e gestione operativa per semplificare la creazione di grandi applicazioni transazionali.

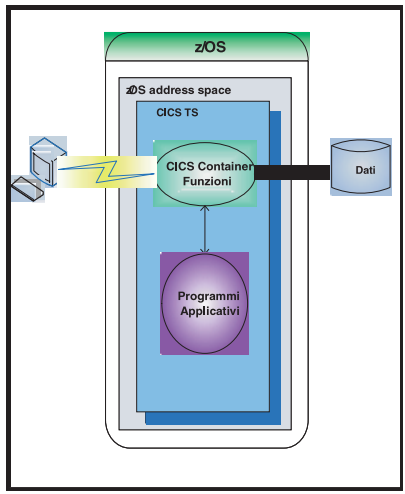


Figura 43 CICS/TM

IMS/TM Information Management/Transaction Manager

L' IMS rappresenta un sistema completo disegnato per alte prestazioni che, oltre ai servizi di base comuni, racchiude al suo interno due componenti fondamentali: una componente di gestione dei dati denominata **IMS/DB**, già analizzata nei paragrafi precedenti e una componente di gestione delle transazioni chiamata **IMS/TM,** di cui andiamo ora ad analizzare alcuni aspetti.

L'IMS/TM è un sottosistema che fornisce l'elaborazione di programmi applicativi che accedono a grandi volumi di dati su Database IMS (DL/I), DB2 o code di messaggi (Message Queue).

Le principali funzioni sono la gestione dei programmi applicativi, il dispatching dei lavori, il caricamento dei programmi applicativi, la gestione del coordinamento e la serializzazione delle risorse interessate. Esso fornisce anche la gestione dei messaggi provenienti dalla rete (3270s, APPC, TCP/IP, WebSphere MQ, ecc.) o da programmi applicativi "batch oriented" non connessi in rete.

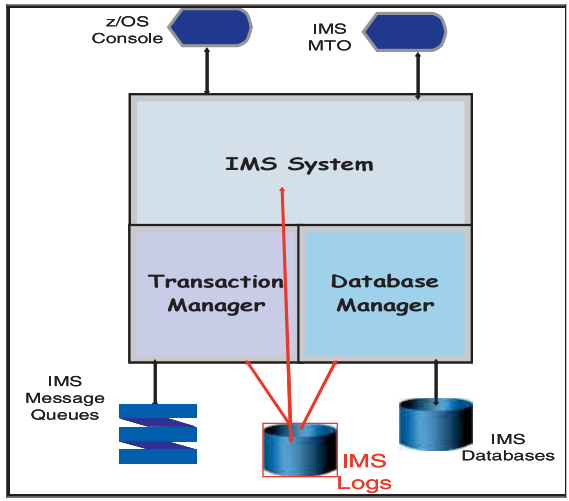


Figura 44 Un sistema IMS

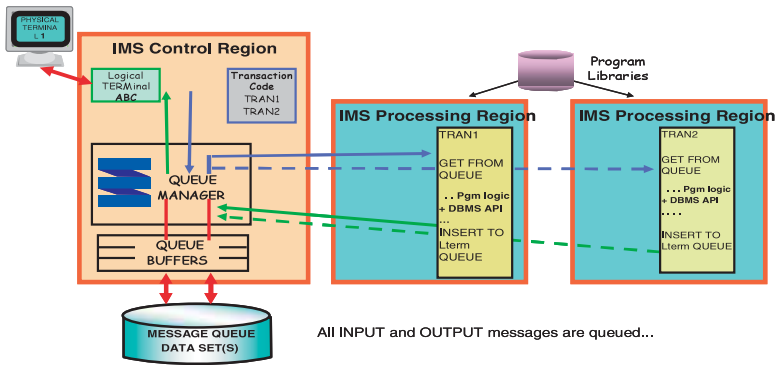


Figura 45 Le regioni IMS

WAS WebSphere Application Server

Il processo di modernizzazione che sta investendo le imprese muove molte applicazioni verso il Web, cosicché nelle attuali realtà ove sono presenti i Sistemi Centrali troviamo lo sviluppo di nuove applicazioni (o la revisione di alcune di quelle vecchie) con tecnologie legate al web e al contempo i workload tradizionali come ad

esempio il lavori batch. In tale ottica la base dei nuovi sottosistemi transazionali è il **WebSphere Application Server** (WAS).

Il WAS è un gestore di applicazioni aderenti allo standard Java 2 Enterprise Edition (J2EE). Esso fornisce inoltre un ambiente di implentazione e runtime costruito su tecnologia standard "open" che supporta tutte le principali funzioni come Servlets, Java Server Pages (JSP) e Enterprise Java Beans (EJB) includendo inoltre le più recenti tecnologie di integrazione di servizi e interfacce.

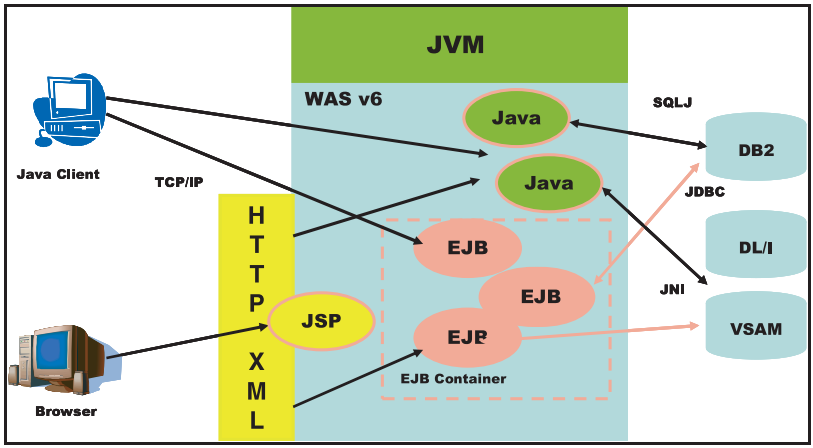


Figura 46 WAS in z/OS

L'ambiente runtime di WAS è altamente integrato con tutte le funzioni e i servizi di z/OS. Infatti esso opera con tutti i maggiori sottosistemi quali DB2, IMS e CICS.
Il WAS in ambiente z/OS enfatizza le caratteristiche di sicurezza, scalabilità e recovery tipiche dei Sistemi Centrali.
La struttura dei processi (Address Spaces) del WAS è costituito da un processo di controllo detto Controller Region (CR) e uno o più processi dipendenti chiamati Servant Region (SR).

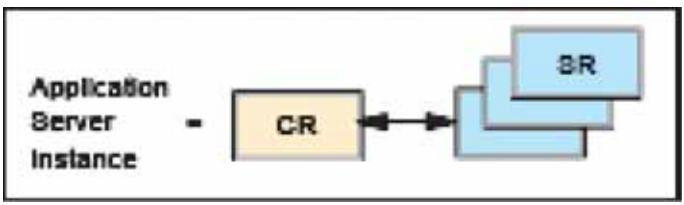


Figura 47 WAS CR & SR

L'application server in z/OS fornisce due possibili configurazioni mantenendo in entrambe le stesse gerarchie architetturali.
Esistono infatti una opzione base (**Base Server Node**) basata su un'unica istanza di application server e di un daemon server e una opzione estesa chiamata **Network Deployment Server** che consente di gestire da un'unica locazione centralizzata più istanze di application server.

3.0.13 Soluzioni Applicative di Business Intelligence (BI)

3.0.13.1 Introduzione

Il Data Warehouse (DW) nasce e si evolve per la necessità di poter prendere decisioni aziendali sulla base di dati di business e di mercato. Nella prima fase negli anni '70 e '80 la focalizzazione era sul come raccogliere e memorizzare grosse quantità di dati che poi venivano utilizzate per supportare le decisioni aziendali. Più tardi, negli anni '90, personal computer, linguaggi di 4° generazione e strumenti di OnLine Analytical Processing (OLAP) spostano la focalizzazione sul come estrarre al meglio le informazioni contenute nei dati per usufruirne in maniera più facile ed immediata. Nasce l'esigenza per l'utente finale di avere un maggiore controllo sui dati ed ogni reparto aziendale vuole una specifica visione o estrazione dei dati per scopi differenti ma originati dalla stessa base, oppure, in alcuni casi, vuole una visione complessiva su specifiche tematiche di business.

Esiste l'esigenza comune di capire le informazioni estratte dai dati e su questa base prendere delle decisioni in maniera veloce, cercando un vantaggio competitivo nel mercato.

Riassumendo, le informazioni in un DW non servono solo per definire le strategie ad alto livello, ma anche per necessità organizzative come il controllo del business, i comportamenti dei clienti, il controllo contro le frodi, l'analisi dei prezzi, delle vendite, della produzione ed altro.

Questi servizi richiedono la possibilità di accedere ai dati globali per integrarli e consolidarli in tempi brevi. Una modalità che comporta una fruizione vicina al 'real time' transazionale con elementi di criticità paragonabili al mondo operazionale.

3.0.13.2 Alcune definizioni

Data Warehouse

Un data warehouse è una organizzazione di dati e/o informazioni con un ampio spettro aziendale il cui utilizzo è alla base per prendere decisioni per supportare i processi e le strategie per una azienda.

Ci sono due tipi di "Data Warehouse Environment":

- l'enterprise data warehouse, che contiene tutti i dati aziendali;
- il data mart, che contiene i dati per una specifica necessità di business.

Data mart

Un data mart (o business process dimensional model) è un insieme di dati orientati ad uno specifica necessità. Ha uno scopo più ristretto e tende ad essere più piccolo in termini di dimensioni. Può contenere sia dati primitivi che sommarizzazioni o dati derivati per rispondere a specifiche esigenze di un gruppo o reparto. Un data mart generalmente supporta una analisi multidimensionale e non necessariamente è basato su un data base relazionale.

Extract, transform, and load (ETL)

ETL è una funzione del data warehouse che gestisce le estrazioni dei dati da varie sorgenti, la trasformazione dei dati nelle informazioni che servono ed il caricamento nel data warehouse e/o data mart. ETL consiste di più processi che sono automatici e paralleli per contenerne la durata. I processi di ETL possono anche eseguire 'la pulizia' dei dati in modo che possano essere trasformati in informazioni consistenti. La trasformazione dei dati può utilizzare aree di transito temporanee durante i processi di estrazione.

Warehouse catalog

Un warehouse catalog è un sottosistema che memorizza e gestisce tutti i metadati ovvero le informazioni sui dati. I metadati contengono informazioni come il mapping tra dato sorgente e di arrivo, il significato del dato, il suo formato ed altro.

Query and reporting

Una query è definita come una domanda di business che è convertita in una computer query. In un processo di query e reporting, la domanda è definita, il dato acceduto e le informazioni trovate e passate alla fase di report che le trasforma in un prospetto o grafico per l'utente finale. I report richiesti possono essere:

- predefiniti - rappresentano un carico stabile;
- ad-hoc - rappresentano un carico variabile.

Online analytical processing (OLAP)

OLAP è un processo di analisi che utilizza una vista multidimensionale dei dati in forma aggregata che permette ottime prestazioni. OLAP normalmente implica l'utilizzo di un multidimensional data store dal quale gli strumenti (come ad esempio Cognos) possono fornire le viste basate sulle dimensioni o sulle serie temporali.

Data mining

Data mining è l'esplorazione e l'analisi di grandi quantità di dati per scoprire informazioni sconosciute o nascoste e relazioni tra dati. Data mining è quindi un processo di scoperta di informazioni, classificazione e previsione.

Business intelligence (BI)

BI è un termine generale per indicare una serie di servizi, tecniche e strumenti a supporto delle decisioni di business basate sulla information technology. BI è uno strato di software che fornisce applicazioni e tecnologie per ottenere reporting, query,

previsioni, dashboarding e analisi del business per gli utenti utilizzando i dati del DW. BI assume che un data warehouse o Data Mart esista con le informazioni corrette, affidabili e con le serie temporali necessarie. Lo strato del software del data warehouse e lo strato applicativo BI permettono di avere un ambiente completo di supporto alle decisioni.

La figura 48 rappresenta una visione schematica dell'ambiente di data warehouse e dello strato di applicazioni BI.

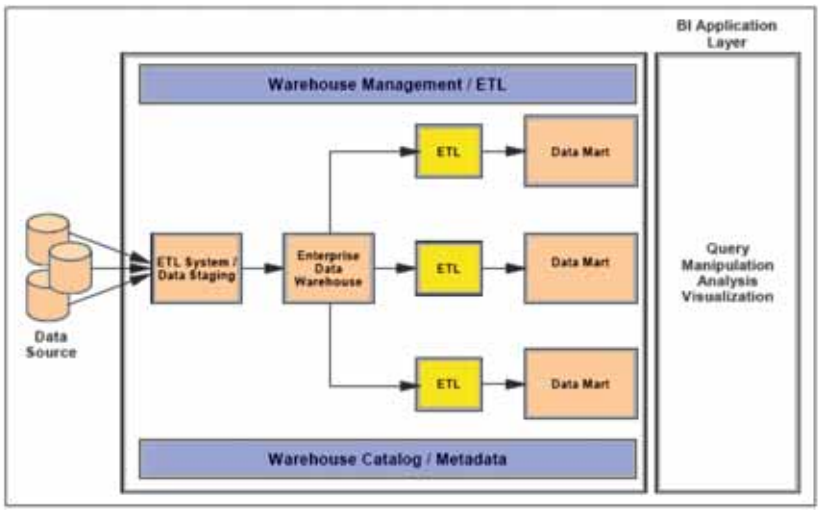


Figura 48 - Ambiente di Data warehouse e di applicazioni BI

In questa introduzione non si vuole entrare nel dettaglio della Business Intelligence, del data modeling e del data analysis che meriterebbero un capitolo ciascuno, ma si vuole dare una idea generale del data warehouse e dei vantaggi che si hanno utilizzando un ambiente mainframe.

3.0.13.3 La Business intelligence nei sistemi z

Nel seguito mostriamo, a titolo di esempio, una possibile architettura per un data warehouse sulla piattaforma System z. I componenti possono essere combinati in varie configurazioni HW e SW, il tutto considerando i volumi di dati da gestire attuali e di tipo storico, il numero di utenti e la loro tipologia.

La figura 49 mostra le opzioni disponibili per alimentare un data warehouse:

- OLTP database → Change Capture → Q Replication → Data warehouse database.
  Si usa per dati che devono essere replicati con minime trasformazioni e un tempo minimo di propagazione in una modalità continua;

- OLTP database → Change Capture → Event Publisher → DataStage → Data warehouse_Database.
  Si usa per propagare specifiche informazioni e/o dati in intervalli predefiniti. DataStage processa i dati catturati e li manipola prima di caricarli nel data warehouse;
- OLTP database → DataStage → Data warehouse database. Prevede un controllo e manipolazione da parte del DataStage, il quale estrae i dati dall'OLTP e li carica nel data warehouse;
- Legacy data → WebSphere Classic Federation → DataStage → Data warehouse_database.
  WebSphere Classic Federation simula l'accesso ai dati tradizionali (legacy), per esempio IMS, VSAM e file sequenziali, come se fossero un relational database management system (RDBMS). Il DataStage legge e processa come richiesto e poi carica i dati nel data warehouse database;
- Distributed data sources → Federation Server → DataStage → Data warehouse database.
  Questa opzione si usa per leggere e/o estrarre dati da dati distribuiti su altre piattaforme sia IBM che di altri fornitori;
- Analisi e produzione di prospetti e/o report sono eseguiti da prodotti come Cognos 8 BI, DataQuant, AlphaBlox o di Terze Parti. Questi prodotti accedono ai dati del warehouse database e generano report, spreadsheet, dashboard, e altro, come richiesto dall'utente finale.

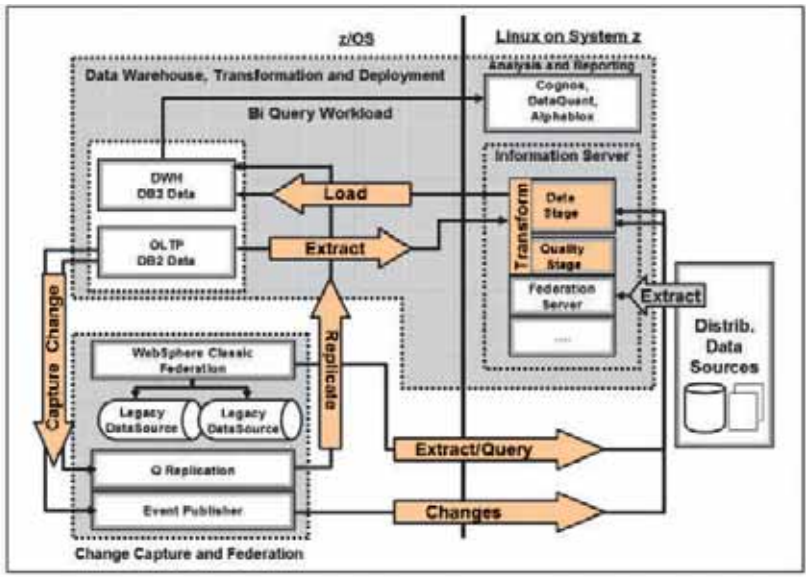


Figura 49 Come alimentare un Data Warehouse

3.0.13.4 Perché realizzare un Data warehouse in ambiente mainframe

Oggi gli ambienti data warehouse sono consolidati in near-real-time operational data stores (ODS) e sono continuamente alimentati da diverse fonti, ad esempio da dati finanziari, di marketing e dagli inventari. Spesso centinaia di utenti accedono al data warehouse con applicazioni di Business intelligence che analizzano e sintetizzano dati per aiutare a prendere decisioni veloci e valide oppure per fare analisi a più ampio spettro. Come risultato si devono fornire diversi livelli di servizio basati sulle diverse criticità e complessità delle richieste. Si deve gestire una combinazione di operazioni critiche ed analitiche in tempi brevi, come le tradizionali operazioni di back-office, di reporting e di analisi per piani strategici e tattici, il tutto complicato da volumi sempre crescenti e dalla storicizzazione dei dati che rendono le richieste più pesanti e sofisticate.

Le soluzioni di BI – Data warehouse devono considerare le esigenze che derivano dalla gestione di un mix delle seguenti condizioni:

- una modalità continua di aggiornamento simile ad un carico OLTP;
- un alto numero di standard report;
- una crescita di richieste ad hoc ed estemporanee;
- una crescita di richieste analitiche e di funzioni di Business intelligence

Il mix di questi quattro diversi tipi di carico è uno degli elementi di differenziazione nella scelta della piattaforma.

La capacità del mainframe di gestire un carico misto lo rendono la piattaforma ideale per queste tipologie di carico. Inoltre vedremo alcune caratteristiche del mainframe che si sposano con un carico BI – Data warehouse.

Disponibilità e scalabilità

La richiesta di disponibilità delle applicazioni in ambiente data warehouse in una soluzione di BI è ormai equivalente tra utenti di sportello e retro sportello. Le applicazioni di BI o data warehouse richiedono lo stesso livello di disponibilità e di sicurezza di ambienti OLTP. La possibilità di incrementare la capacità sia del DB2 per mainframe che della piattaforma HW permettono una scalabilità unica.

Workload management

Il Workload Manager (WLM) ha dato ampia prova nella realtà dell'utilizzo quotidiano di poter gestire carichi diversi contemporaneamente massimizzando l'uso delle risorse disponibili. Con il WLM, una unità di lavoro può avere una priorità iniziale basata sull'importanza in termini di business. Questa nel tempo può essere modificata in base alle necessità, come codificato nelle politiche del WLM riguardo al DB2 per z/OS. Il DB2 lavora in stretta cooperazione con il WLM per assicurare che le priorità e le eventuali alterazioni siano applicate con effetto immediato sulla query in esecuzione.

Compressione dei dati

Il DB2 su mainframe comprime i dati in modo da ridurre drasticamente sia l'occupazione su disco che il tempo delle elaborazioni. La compressione e la decompressione dei dati è ottenuta usando una speciale assist HW presente sui mainframe dal 1993: la Compression Assist Feature, che rende le operazioni estremamente veloci e poco costose. Con una percentuale del 50% di compressione una pagina di dati contiene il doppio dei dati e questo significa che ogni I/O fornisce il doppio dei dati rispetto a dati non compressi. I dati rimangono compressi nei buffer pool in memoria e quindi anche qui abbiamo il doppio dei dati memorizzati. Quando i dati sono modificati le informazioni riportate sul log a loro volta sono compresse, riducendo il volume dei log del DBMS.

Regolamenti di sicurezza

I regolamenti come Sarbanes-Oxley Act (SOX), Basel II, Data Protection Act (UK) e U.S. Patriot Act sono stati creati per proteggere gli investimenti ed evitare frodi. Le aziende devono attenersi a questi regolamenti e devono assicurarsi che i dati siano sicuri, l'accesso sia controllato, le modifiche siano sorvegliate ed esista una modalità di disaster recovery. Il mainframe fornisce i più alti livelli di certificazione di sicurezza, le funzioni di crittografia HW e le funzioni per prevenire intrusioni.

Disaster Recovery

Con il crescere dell'importanza della BI e dell'importanza strategica del Data warehouse, le richieste di un Disaster recovery per questi ambienti sono simili alle esigenze degli ambienti operazionali. Quindi è essenziale considerare gli scenari di Disaster recovery prima di una realizzazione di una soluzione di data warehouse.
Le seguenti tecnologie per ambienti mainframe e DB2 for z/OS forniscono le migliori soluzioni disponibili sul mercato:

- Copy Services
- GDPS/XRC
- GDPS/PPRC
- GDPS/PPRC HyperSwap™
- BACKUP and RESTORE SYSTEM utilities of DB2 for z/OS

Utilizzo del processore specializzato zIIP

Anche in questo ambito si devono considerare i benefici introdotti dalla continua innovazione che riducono costantemente il Total Cost of Ownership (TCO). L' IBM z10 Integrated Information Processor (zIIP) viene utilizzato nel DB2 V8 e V9 for z/OS per le richieste distribuite via TCP/IP, per le query che permettono l'utilizzo del parallelismo per la loro risoluzione e per la parte relativa agli indici di alcune utilities. Le applicazioni di BI sono le candidate ideali per poter beneficiare dell'utilizzo di questi processori specializzati. La figura 50 illustra i vantaggi dell'utilizzo dei motori zIIP.

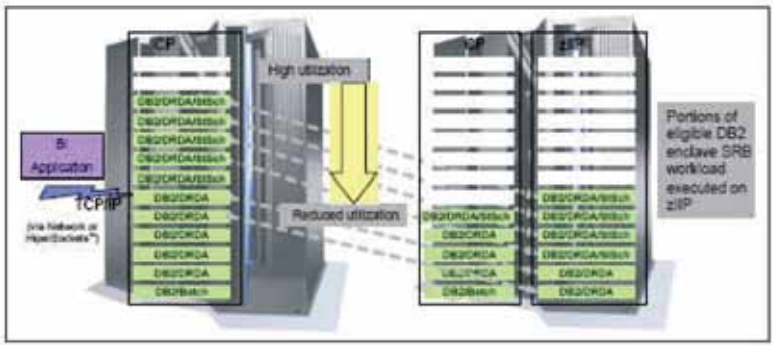


Figura 50 Utilizzo processore specializzato zIIP

3.0.14 z/OS Management Facility

L'utilizzo di alcuni tool di sviluppo e gestione di z/OS può risultare ostico, soprattutto per chi si affaccia per la prima volta al mondo mainframe.

Negli ultimi anni IBM ha investito molto per cercare di semplificare l'utilizzo di questi strumenti: RDz - *Rational Developer for System z* - e *IBM Health Checker for z/OS* sono due esempi a dimostrazione della strada intrapresa in questi anni.
Sempre in linea con questa tendenza, nel settembre 2009 è stato annunciato *z/OS MF release 1.11 - zOS Management Facility* - che grazie alla sua *Web browser user interface* facilita la gestione dell'ambiente mainframe.
Un'interfaccia *user-friendly* consente non solo ai nuovi utenti di acquisire rapidamente gli *skills* necessari all'amministrazione del sistema, ma è utile anche agli utenti più esperti per eseguire alcuni *task* in tempi più rapidi.
IBM z/OS Management Facility release 1.11, nonostante il nome che si riferisce alla versione del sistema operativo, è alla sua prima release e nonostante ad oggi implementi solo delle funzioni limitate, è destinata a diventare la console principale di amministrazione del sistema z/OS, facile e intuitiva grazie alla sua interfaccia *java-based*.
L'idea alla base di questo nuovo prodotto è dunque quella di semplificare l'approccio a z/OS e ridurre i tempi di alcune attività, centralizzando tutta la gestione in un unico ambiente web.
Come mostrato in figura 51, nella prima versione z/OS MF si focalizza su due aree che sono fra le più delicate della gestione di un sistema z/OS:

- *Problem management*, implementando delle funzioni che offrono la possibilità di reperire facilmente le informazioni a seguito del verificarsi di un errore;
- *Configuration management*, dando la possibilità di impostare velocemente le configurazione di rete, in particolare per quanto riguarda il *Communication Server di z/OS.*

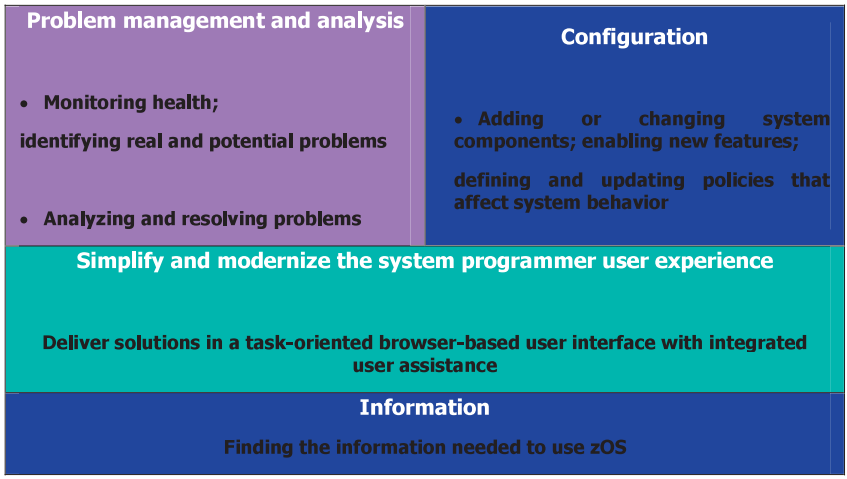


Figura 51 Obiettivi principali della prima release di z/OS MF

I futuri obiettivi che verranno implementati nelle versioni successive di z/OSMF riguardano principalmente la *Security Administration,* il *Network Admistration,* lo *Storage Management,* e il *Workload Management.*

Come mostrato in figura 52, il prodotto viene installato in z/OS e si appoggia ad una speciale versione di *WebSphere:* per questo motivo, z/OS MF è fornito assieme a *Websphere Application server OEM Edition V7.0 for z/OS*.

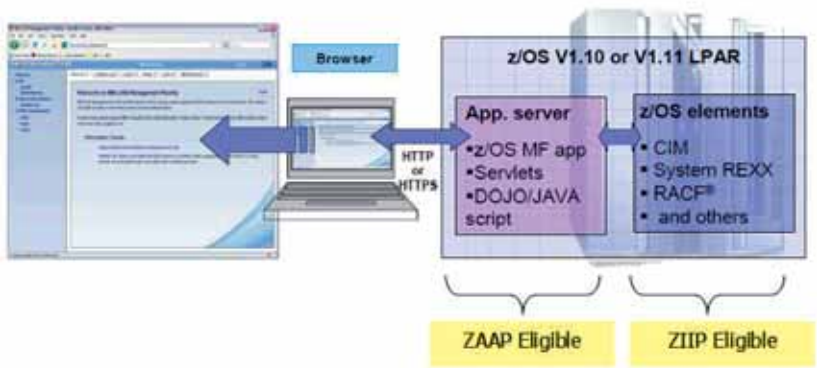


Figura 52 z/OS MF: Struttura interna

z/OS MF è una applicazione di z/OS con un accesso diretto a dati e informazioni di z/OS, e con un'interfaccia browser per le workstation.
Come mostrato in figura 52, una parte di z/OS MF utilizza la *DOJO technology* per la GUI- *Graphical User Interface*-.
Tutto dunque è installato sul server z/OS, senza bisogno di particolari requisiti e installazioni lato client (la comunicazione tra workstation e z/OS MF avviene attraverso HTTPS).
La z/OS MF interroga lo z/OS, acquisisce le informazioni, e attraverso l'interfaccia web le mostra all'utente in maniera semplice e veloce.
In figura 53 viene riportata la pagina principale dello z/OS MF accessibile via web: sulla sinistra è possibile accedere al menu principale, composto da quattro voci.

- *Configuration:*

La prima voce in elenco riguarda il *Configuration Assistant* per lo *z/OS Communication Server* che permette di semplificare la configurazione e il setup della rete TCP-IP.

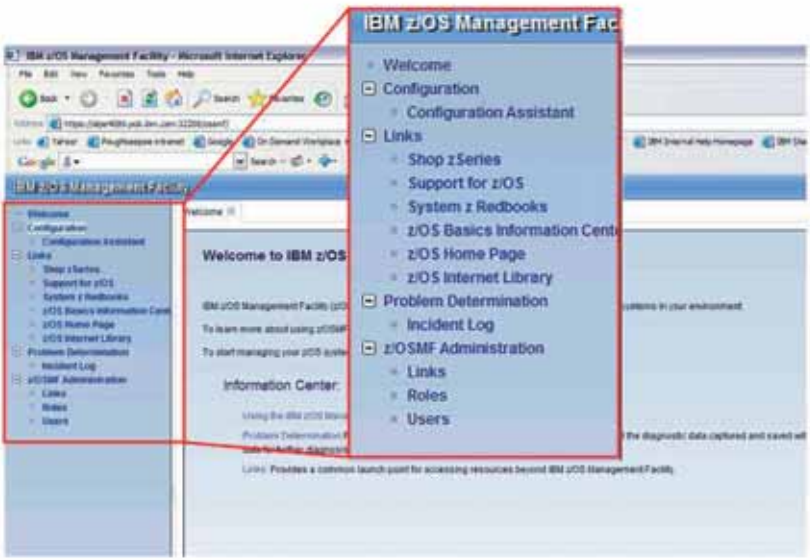


Figura 53 Dettaglio del menu della Welcome page dello z/OS MF

La configurazione è tipicamente un'operazione che richiede vari step per definire i file di configurazione, le procedure di start, e le policies di sicurezza. IBM Configuration Assistant for z/OS Communication server, che prima esisteva come tool per WINDOWS, fornisce una GUI di configurazione che può essere usata per generare e gestire i file di configurazione e le policies per *Application Transparent-Transport Layer Security (AT-TLS), IP Security (IPSec), Network Security Services (NSS), Policy Based Routing (PBR), Quality of Service (QoS),* e *Intrusion Detection Services (IDS).*

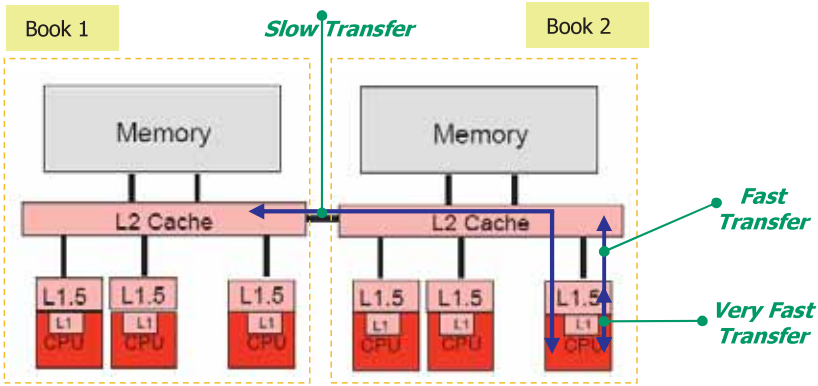


Figura 54 Struttura hardware dei livelli di memoria in un ambiente multi-book

Nella figura 55 vengono riportati i tempi medi di accesso ai vari livelli di memoria nel system z10; L2 e RAM sono considerati *remoti* (*non-local)* se sono posizionati su un book diverso da quello nel quale risiede il processore.

	Cycles	nsec
L 1.5	13	2.9
L2	88	20
L2-non Local	229	52
RAM Local	591	134,3
RAM non Local	712	161,8

Figura 55 Numero di cicli e tempi di accesso alla memoria

Per valutare i numeri della tabella possiamo assumere che l'accesso alla cache L1 sia di pochi cicli (< 1 nsec). Si capisce subito che quando più distante è il livello di memoria da cui il dato deve essere prelevato, tanto maggiore diventa il tempo di esecuzione delle singole istruzioni: diventa perciò imperativo massimizzare lo Hit rate nei vari livelli della cache.

Negli ultimi anni è stato fatto un grande lavoro su z/OS e sul System z10 per ottimizzare l'uso della cache cercando di aumentare il *cache-hit ratio.* A fronte di questo, una delle principali novità introdotte è l'algoritmo di Hiperdispatch.

Prima dell'introduzione dell'Hiperdispatch sia lo z/OS che il PR/SM erano poco sensibili alla topologia dei processori sui quali le *Unit of Work* -UOW- venivano eseguite: un modello di questo tipo è definito *Horizontal Dispatch mode* e si contrappone al *Vertical Dispatch mode* su cui si basa invece l'algoritmo di Hiperdispatch.

Il lavoro è organizzato in un'unica coda di dispatch e a tutti i processori logici - *Logical Processor (LP)* - definiti all'interno della partizione logica viene assegnato lo stesso *logical processor share*, definito come la percentuale di tempo in cui un processore logico ha diritto ad utilizzare il processore fisico - *Phisical Processor PP*. Nel dispatch si hanno due fasi distinte; la prima nella quale le UOW vengono mandate in esecuzione su un processore logico (fase eseguita dallo z/OS), che a sua volta viene eseguito sul processore fisico in accordo con il logical processor share definito (seconda fase, eseguita dal PR/SM, ovvero l'hypervisor che da un punto di vista architetturale è considerato parte HW).

Lo z/OS manda in esecuzione le UOW su un qualsiasi processore logico in stato di *ready*, senza preoccuparsi di dove la UOW fosse stata mandata in esecuzione nel dispatching precedente. Nel secondo step del dispatch, il PR/SM manda in esecuzione il processore logico su un qualunque processore fisico ma senza creare una stretta corrispondenza tra i due.

Tutto questo aumenta la probabilità di generare un *cache miss* aumentando il tempo di accesso alla memoria e penalizzando le prestazioni: infatti, se una unit of work interrotta durante la sua esecuzione (per esempio a seguito di una richiesta di I/O) una volta ripresa venisse mandata in esecuzione su un processore fisico diverso da quello in cui era in esecuzione prima dell'interruzione, non sfrutterebbe la stessa cache generando un cache miss e quindi aumentando la probabilità di dover accedere ad un livello di cache remota.

Un miglioramento sostanziale delle prestazioni si è avuto con l'introduzione dell'algoritmo di Hiperdispatch, che prevede il *re-dispatching* sullo stesso processore delle Unit of Work interrotte, massimizzando in questo modo il cache-hit ratio. Come detto, redistribuire un task che viene interrotto sullo stesso processore fisico (o sullo stesso nodo) permette infatti di sfruttare la stessa cache.

Comunque, dato che i processori del systemZ hanno un altissimo livello di utilizzo, il tentativo di *re-dispatch* della UOW sullo stesso processore potrebbe creare un ritardo superiore ai benefici attesi: per prevenire questo inconveniente, l'algoritmo dell'Hiperdispatch prevede di mantere le UOW su uno stesso gruppo di processori (*nodo o, in termini architetturali, Logical Processor affinity pool*) situati sullo stesso book, ovvero che condivide la stessa cache locale L2.

Per mantenere la UOW sullo stesso processore (o gruppo di processori) per l'intera vita della UOW stessa, è necessario però creare una forte corrispondenza (affinità) tra UOW-LP-PP.

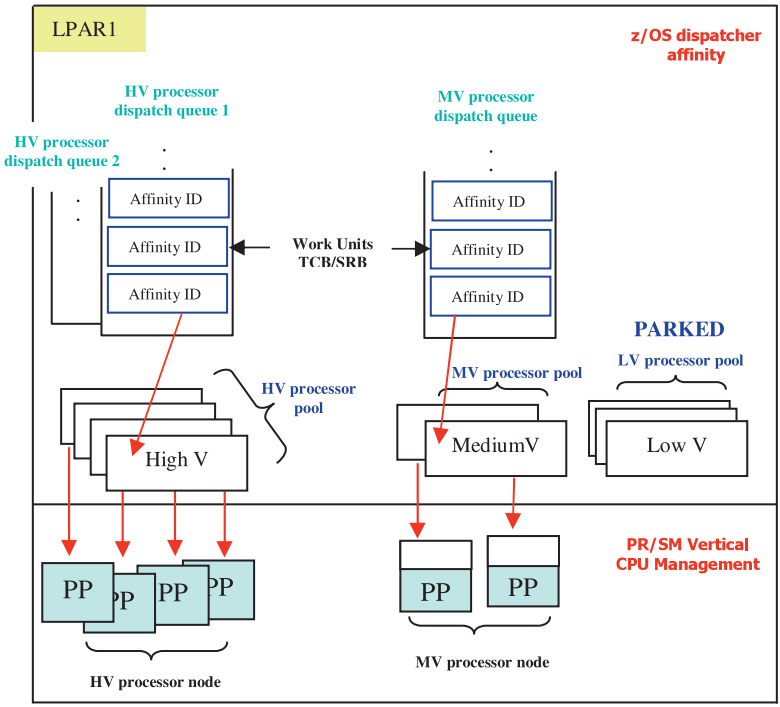


Figura 56 Gestione verticale nell'Hiperdispatch

Come mostrato in figura 56 ci sono due componenti principali che intervengono per gestire la modalità Hiperdispatch creando delle affinità tra le risorse: lo *zOS dispatcher affinity* e il *PR/SM Vertical CPU Management*.
Queste due componenti cooperano scambiandosi informazioni per ottimizzare al meglio le risorse disponibili della LPAR che lavora in modalità Hiperdispatch.
Lo z/OS dispatcher affinity crea delle *multiple affinity dispatch queues:* a ciascuna UOW viene assegnato un *affinity identifier* e le UOW con le stesso affinity identifier verranno mandate nella stessa affinity dispatch queue.
Da due a cinque processori logici vengono raggruppati creando dei logical *processor affinity pool* che condividono la stessa affinity dispatch queue.
L'affinity identifier assicura dunque che una stessa UOW venga mandata in esecuzione sempre sullo stesso gruppo di LP creando una affinità tra la UOW e un gruppo di processori logici. z/OS cercherà comunque di associare alla stessa UOW sempre lo stesso processore logico, ma se, trascorso un tempo "ragionevole", questo non fosse possibile, assegnerà un processore logico appartenente allo stesso pool.
Il PR/SM provvederà a mappare *one-to-one* tutti i processori logici dello stesso pool su dei processori fisici nello stesso book (creando dei *Phisical processor Node*, che vengono dunque definiti in modo da contenere processori sullo stesso book).

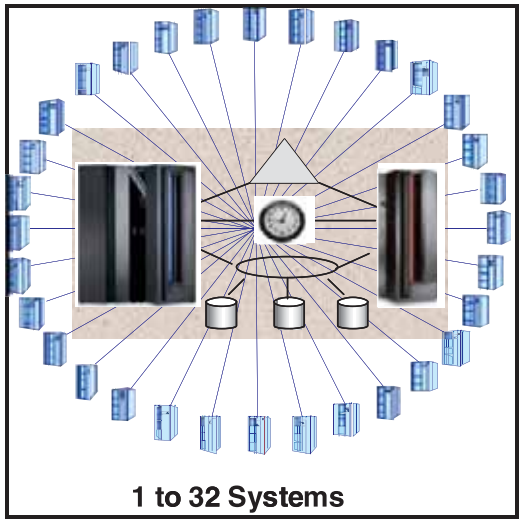


Figura 57 Parallel Sysplex

Per realizzare la condivisione dei dati e il controllo centralizzato è necessario disporre di un terzo componente, in genere duplicato, denominato **Coupling Facility** (vedi figura 58). Coupling Facility è una partizione logica governata da un sistema operativo specializzato detto Coupling Facility Control Code (CFCC). In esso la memoria interna è dinamicamente partizionata in entità dette "strutture", che hanno funzioni specifiche e posso essere di 3 tipi:

- **Cache:** usata come un buffer ad alta velocità per memorizzare dati acceduti in read e/o write da tutti i componenti del cluster. Viene usato un meccanismo di "buffer invalidation" per garantire Read e Write integrity alle applicazioni che usano i dati in cache
- **List:** usato per funzioni di condivisione di aree di lavoro e per memorizzare lo status riguardo a funzioni e/o componenti di sistema. Funzione essenziale è lo scambio di messaggi fra membri del SYSPLEX
- **Lock:** per serializzare l'accesso alle strutture condivise.

Per connettere tra loro le componenti di un Parallel Sysplex si usano connessioni ad alta velocità dette Coupling Links e per far sì che i Clock di tutte le immagini componenti il Cluster siano sincronizzati tra loro viene utilizzata una funzione dei Mainframe chiamata Server Time Protocol (STP). L'utilizzo di questa funzione permette di avere due immagini connesse nello stesso Parallel SYSPLEX collocate ad una distanza massima di 100 Km.

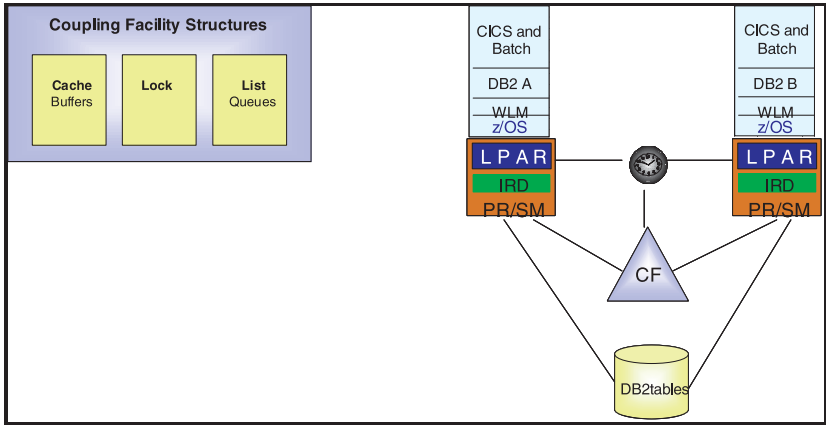


Figura 58 Coupling Facility

Le funzioni fondamentali del Parallel Sysplex sono le seguenti:

- **Data Sharing:** condivisione di dati gestiti da sottosistemi quali DB2, IMS, VSAM, con garanzia della integrità in read e write. L'accesso ai dati può avvenire anche da sistemi posti in locazioni fisicamente lontane (purché all'interno del Parallel SYSPLEX).
- **Operazioni centralizzate:** Ogni console collegata ad un'immagine del SYSPLEX può inviare comandi ad ogni altra immagine del SYSPLEX, così come può ricevere messaggi da qualunque immagine: questa funzione è anche chiamata SYSPLEX Console.
- **Scalabilità orizzontale:** i sistemi possono essere aggiunti e attivati senza interruzione del servizio e in maniera trasparente alle applicazioni.
- **Bilanciamento Automatico del carico:** ciascun carico di lavoro sarà eseguito con un obiettivo di prestazioni pre-assegnato in base alle politiche del Workload Manager e potrà essere dirottato in maniera automatica e trasparente sul sistema meglio posizionato degli altri per quanto riguarda l'occupazione di carico ed utilizzo di risorse, cioè in grado di ricevere altro lavoro.
- **Disponibilità Continua**: il cluster continua a fornire servizio nel caso una delle sue componenti dovesse fermarsi o cadere. Per questa ragione si presenta come un unico sistema in grado di assicurare la disponibilità continua di dati e applicazioni.

3.0.17 UNIX System Services

USS, *z/OS UNIX System Services*, o come spesso è abbreviato *z/OS UNIX*, è un ambiente operativo UNIX implementato all'interno del sistema operativo z/OS.

Il lavoro di implementazione di UNIX all'interno di z/OS è iniziato all'inizio degli anni '90: la prima versione di z/OS UNIX era chiamata OpenEdition (OE o OMVS), poi diventata OS/390 UNIX System Services, e infine z/OS UNIX System Services, nome

con cui la conosciamo ancora oggi. USS è una implementazione certificata UNIX ottimizzata per il mainframe, quindi permette alle applicazioni UNIX di altre piattaforme di poter essere eseguite su mainframe, previa ricompilazione.

I programmi che usano USS hanno accesso completo e sicuro alle funzioni interne di z/OS: applicazioni come WebSphere Application Server, CICS, IMS, DB2, Lotus Domino e molte altre usano USS.

z/OS ha quindi la capacità di mostrarsi all'utente in due maniere diverse a seconda delle esigenze, come ambiente UNIX o come ambiente tradizionale MVS.

Interazione con gli elementi di MVS

Come accennato, z/OS UNIX interagisce con vari elementi di z/OS. Vediamo quali sono le principali componenti z/OS con cui z/OS UNIX interagisce:

- WLM, che consente di gestire l'allocazione delle risorse fisiche, massimizzando le prestazioni
- SMF, *System management facilities*
- Compilatori C/C++
- *Language Environment*, per eseguire shell e utilities
- DFSMS -*Data Facility Storage Management Subsystem* (La gestione dei dataset zFS è affidata al DFSMS)
- *Security Server* di z/OS, per gestire gli accessi e sfruttare le funzioni di RACF, componente del Security Server
- RMF - *Resource Measurement Facility*, che colleziona i dati per controllare performance e livelli di servizio.
- SDSF - *System Display & Search Facility*, per la gestione dei job
- TSO/E - *Time Sharing Option Extensions*, che permette di interoperare tramite vari comandi, (il commando OMVS permette di entrare nella shell UNIX)
- *z/OS Communications Server*, che gestisce i servizi di rete TCP/IP, come Telnet e rLogin
- TSO/ISPF, per la ISHELL
- Network File System (NFS)
- z/OS Distributed File Service

Come si nota dall'elenco sopra riportato, UNIX System Services è componente strettamente integrato all'interno del sistema operativo z/OS.

Per essere aderente allo standard UNIX lo z/OS deve poter gestire anche i file UNIX organizzati in Hierarchical File System (HFS). L'HFS è creato all'interno di speciali Data set: zSeries File system (zFS), un cluster VSAM di tipo Linear.

L'insieme degli zFS Dataset collegati con la tecnica dei Mount Point costituisce lo z/OS UNIX file System (vedi Figura 60). Le applicazioni (classiche o UNIX) possono lavorare con dati sia dello *z/OS UNIX file systems* sia dei tradizionali *MVS dataset*.

I parallelismi tra l'ambiente MVS e l'ambiente shell

La figura 59 mostra alcune funzioni base dell'MVS e l'equivalente (o simile) implementazione in z/OS UNIX.

Function	MVS	UNIX
Background work	Submit batch JCL	Command &
Configuration parameters	SYS1.PARMLIB	/etc
Data management	DFSMShsm	tar, cpio, pax
Editor	ISPF option 2	oedit, ishell
Initiate new task	ATTACH, LINK, XCTL	fork(), spawn()
Interactive access	Logon to TSO	telnet/rlogin to sh/tcsh
Job management	SDSF	ps, kill
List files	ISPF option 3,4	ls
Long running work	Started task (STC)	daemon
Power user	RACF OPERATIONS	superuser or root
Primary data index	Master Catalog	root ("/") directory
Procedural language	CLIST, REXX	shell scripts
User identity	user/group	UID/GID

Figura 59 Equivalenza delle funzioni di MVS e z/OS UNIX

In UNIX, i parametri di configurazione sono tipicamente salvati nella directory /etc, e in maniera analoga si trovano nella libreria SYS1.PARMLIB del sistema MVS.
Un daemon UNIX è un programma che gira continuamente in background con lo scopo di gestire periodicamente dei servizi e delle richieste, similmente a quanto fanno in MVS gli Started Task.
La gestione della memoria virtuale in ambiente UNIX è totalmente demandata al meccanismo classico dello z/OS, con l'aggiunta di una particolare funzione (Copy On Write – COW) implementata per velocizzare l'esecuzione della FORK UNIX.
La shell è l'interprete dei comandi, che gli utenti UNIX incontrano durante la fase di login, del tutto simile alla line-mode della sessione TSO.

z/OS UNIX è diverso dalle altre implementazioni UNIX, dato che non è un'entità indipendente come siamo soliti vedere su altri sistemi. Per questo motivo, bisognerà avere la possibilità di poter definire "dall'esterno", ossia da z/OS, l'ambiente di lavoro e il file system UNIX, fase che prende per l'appunto il nome di configurazione esterna: per far questo si utilizza il membro BPXPRMxx di SYS1.PARMLIB.

La configurazione interna di z/OS UNIX, è invece definita nello stesso modo di un qualsiasi classico sitema UNIX, attraverso i file di configurazione nella directory /etc.

Organizzazione del file system

In figura 60 è mostrato un esempio dell'organizzazione dei file in ambiente USS/MVS.

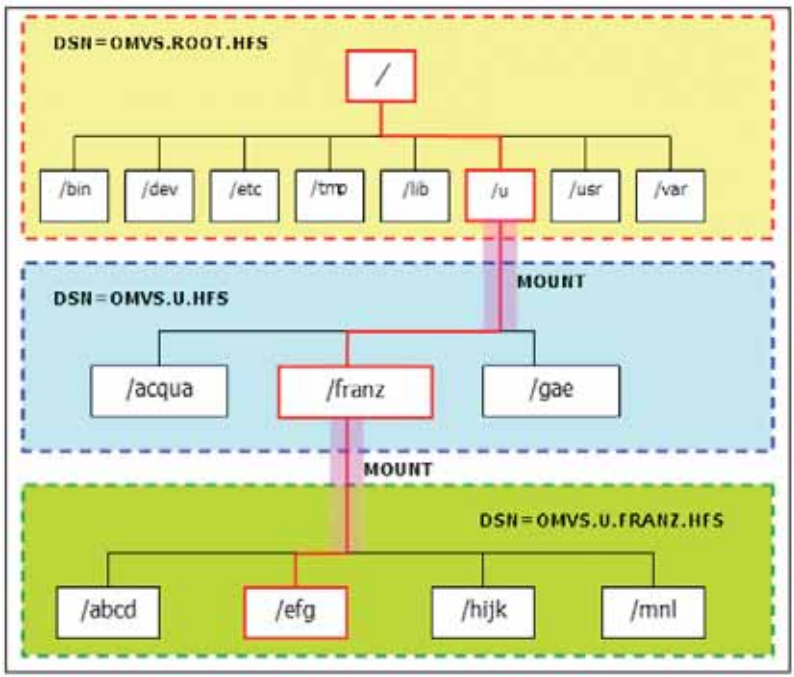


Figura 60 Organizzazione dello z/OS UNIX file system

I file di USS sono organizzati a gerarchie di livelli, come in tutti i sistemi UNIX e a differenza di z/OS. Tutti i file appartengono a una certa directory, che a sua volta appartiene ad un altra e così via fino al livello più alto, la root directory.

Dal punto di vista di MVS, un'intera gerarchia di file è una collezione di dataset zFS (zSeries File System che supporta l'ACLs), gestibili attraverso il componente DFSMS.

Ogni dataset zFS ospita almeno un file system "montabile".

Il root file system è il primo file system montato, dopo di che si possono montare gli altri file system sotto qualsiasi altra directory dentro il root file system o sotto una directory già montata.

3.1 La Virtualizzazione Software – z/VM

Questo capitolo descrive il sistema operativo z/VM, esempio di virtualizzatore Software basato sulla z/Architecture. Di esso descriveremo la gestione delle risorse fisiche e le modalità di attivazione delle "Macchine Virtuali". Rimandiamo al capitolo su zLinux la descrizione dell'implementazione di un ambiente virtualizzato z/VM con macchine virtuali zLinux.

3.1.0 Il Sistema Operativo z/VM - Caratteristiche

In tema di virtualizzazione non si può dimenticare il primo Virtualizzatore Software sviluppato da IBM, denominato Virtual Machine (VM). La sua versione attuale aggiornata alla z/Architecture è denominata z/VM.

Questo sistema operativo, nato alla fine degli anni 60, prima dei virtualizzatori hardware, si basa sui concetti di memoria virtuale per dividere le risorse di un singolo computer in una molteplicità di computer virtuali indipendenti e isolati, ognuno dei quali dotato di potenza elaborativa, memoria centrale, memoria secondaria (dischi, nastri) e altri dispositivi I/O ed ognuno con il suo proprio sistema operativo. Le risorse, dal loro canto, sono gestite da uno strato software e aggregate in insiemi condivisi. Esse sono attribuite agli utenti come risorse virtuali, separando la presentazione delle risorse dalle entità fisiche reali. Su questi computer virtuali operano sistemi operativi di vario tipo (CMS, z/OS, z/Linux , z/VM ed altri) aderenti anch'essi alle varie architetture supportate (es XA, ESA/390,...), compresa la z/Architecture.

Ogni sistema operativo ospite ha a disposizione un ambiente elaborativo completo, del tutto analogo ad un ambiente fisico. I servizi di virtualizzazione di z/VM fanno credere al sistema operativo ospite di avere il completo controllo dei processori e delle altre risorse loro assegnate. Dal suo ruolo di controllo z/VM inoltre può far condividere le risorse fisiche delle varie CPU, devices e network alle varie immagini di Sistemi ospite come nella figura 1.

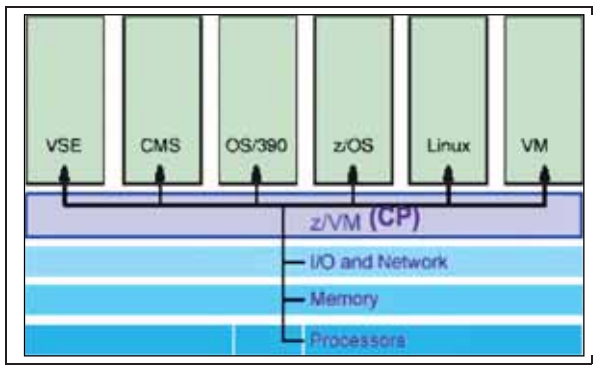


Figura 1 Condivisione risorse - ambiente z/VM

Gruppi di macchine virtuali possono essere collegate insieme e formare network virtuali. Le funzioni fondamentali che un virtualizzatore e quindi anche z/VM deve assicurare sono:

- **Multitasking:** ovvero supportare l'esecuzione contemporanea di più sistemi operativi ospiti (guest OS).
- **Isolamento:** cioè garantire che il malfunzionamento di una macchina virtuale e del sistema operativo su essa operante non si ripercuota sulle altre macchine virtuali.
- **Gestione delle risorse e del carico elaborativo:** ovvero non deve accadere che un sistema operativo si accaparri tutta la potenza di calcolo della CPU rallentando quasi completamente gli altri, né che possa allocare tutta la memoria disponibile privandone gli altri.
- **Amministrazione:** L'amministratore di sistema deve poter avviare, interrompere, riconfigurare o clonare le macchine virtuali senza dover arrestare quelle non interessate dall'operazione.

z/VM fornisce una gestione centralizzata dei dati e del loro backup e inoltre ha funzioni di monitoraggio del carico e delle performance. Il vantaggio legato alla virtualizzazione e alla condivisione delle CPU virtuali è il cosiddetto "over commitment" vale a dire la possibilità di definire risorse virtuali che sono sovrabbondanti rispetto a quelle fisiche. Infatti, se una macchina non può utilizzare tutta una risorsa a lei assegnata, la parte da essa non utilizzata può essere disponibile ad altre risorse.

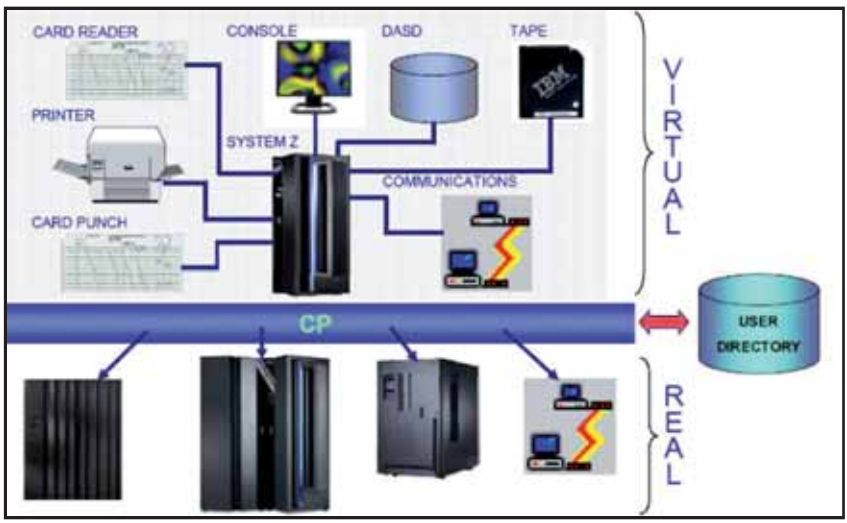


Figura 2 Ambiente z/VM

Le peculiarità di z/VM in termini di gestione delle risorse (ovvero di suddivisione di un sistema singolo in sistemi virtuali multipli ognuno dotato di risorse proprie) lo rende particolarmente utile nel workload interattivo, nelle applicazioni client/server e nel supporto ai sistemi operativi. Infatti nel caso di elaborazione interattiva il sistema operativo z/VM fornisce una base per un dialogo flessibile tra utenti e programmi applicativi; nel caso di client/server è molto comune la configurazione in cui una macchina ospita un programma che fornisce servizi (server) a utenti di altre macchine virtuali (clients); le efficienze generate attraverso z/VM lo rendono adatto a creare, testare ed eseguire applicazioni per i vari ambienti operativi supportati.

Il modello elaborativo di z/VM utilizza un programma per gestire le risorse hardware di un computer. Tale programma è chiamato Control Program (CP). Le attività degli utenti (macchine virtuali) sono gestite da un sistema operativo ospite. Un particolare sistema operativo fornito insieme a z/VM si chiama Conversational Monitor System (CMS).

3.1.1 Le Macchine Virtuali

Il termine Macchina Virtuale, in questo contesto, viene usato per indicare un ambiente di esecuzione che si comporta come se fosse un computer a se stante ma che in realtà condivide con altre macchine virtuali la stessa macchina fisica.

La "macchina virtuale " è completamente indipendente e lavora in contemporanea con le altre macchine virtuali sotto il controllo di z/VM. Questo è in grado di creare "macchine virtuali" tra loro indipendenti che possono usare le risorse interne del computer (memoria, dischi, porte di comunicazione) senza andare in conflitto tra loro.
La tecnologia di Virtualizzazione è quindi trasparente al Sistema Ospite. La gestione delle macchine virtuali presenta molti aspetti, quali i meccanismi con cui viene gestita la Memoria Virtuale di z/VM, la gestione di risorse condivise (CPU, memoria, dispositivi di I/O), la gestione delle operazioni di I/O, ecc.

Un Sistema Operativo che opera su una macchina virtuale ha la stessa operatività di quello che gira su una macchina reale. Ogni dispositivo assegnato ad una macchina virtuale, come la memoria, il processore o un qualsiasi dispositivo di I/O, si deve comportare come se la macchina virtuale fosse reale. Nuovi sistemi ospiti posso essere aggiunti velocemente senza richiedere risorse dedicate.

Ci sono vari sistemi operativi ospiti supportati come ad esempio z/OS, zLinux, z/VSE, z/TPF ed anche z/VM di secondo livello (vedi figura3).

Come accennato precedentemente, z/VM permette di far funzionare copie multiple e tipi differenti di Sistemi Operativi sullo stesso Sistema Centrale.

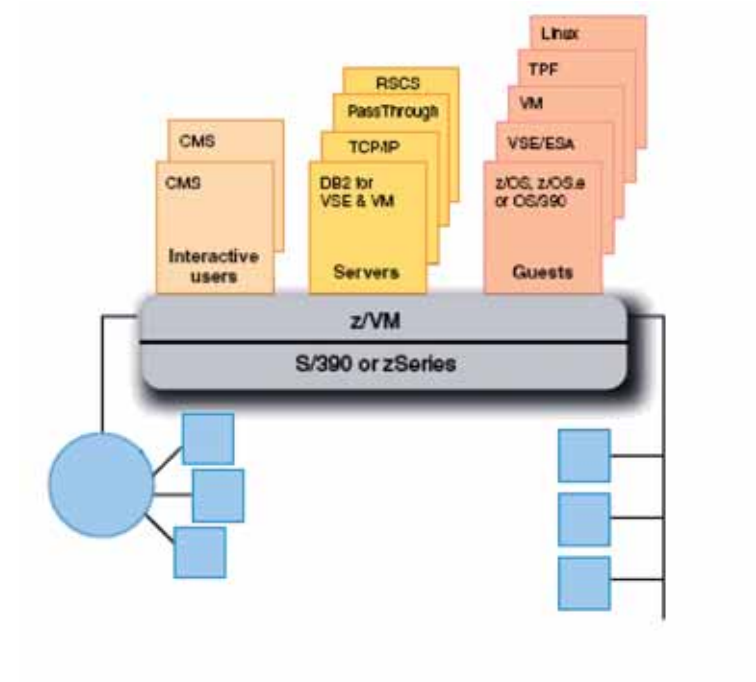


Figura 3 Sistemi operativi ospiti - ambiente z/VM

z/VM tramite uno dei suoi componenti fondamentali, il CP, ha un file di configurazione fondamentale denominato "SYSTEM CONFIG" che definisce, tra le altre cose, le risorse reali a sua disposizione che poi dovrà gestire. In particolare le informazioni sui dischi su cui risiederà il sistema operativo z/VM, la lista dei dischi che utilizzerà il CP, i dischi a disposizone delle macchine virtuali, la configurazione di memoria reale e la configurazione di tutti gli altri dispositivi reali.

D'altro canto z/VM mantiene la descrizione di tutte le informazioni delle macchine virtuali ospitate nel sistema anche nel file USER DIRECT chiamato z/VM User Directory; tale file risiede in memoria alla attivazione di z/VM. All'interno di un sistema z/VM si possono considerare vari tipi di Macchine Virtuali:

- **Macchine virtuali mono-utente con Sistema Operativo CMS:** Macchine Virtuali che ospitano un particolare Sistema Operativo mono-utente, denominato Conversational Monitor System (CMS). Sotto CMS si possono avere ambienti di produttività personale o di sviluppo delle applicazioni o della istallazione e manutenzione di z/VM

- **Macchine Virtuali con Sistemi Ospiti:** Macchine Virtuali che ospitano un qualsiasi Sistema Operativo a 31-bit e 64-bit che usa una architettura supportata tra cui la z/Architecture; tra queste possiamo annoverare macchine che ospitano Sistemi z/VM di Secondo livello ovvero Macchine

Virtuali che contengono una nuova istanza di z/VM, utilizzate a vari scopi (manutenzione, nuove generazioni, ecc.)

- **Macchine Virtuali di Servizio:** Macchine CMS con applicazioni di servizio per tutto l'ambiente z/VM cui si rivolgono gli utenti delle altre macchine virtuali e del CP per usufruire di particolari servizi. Esempi di Macchine Virtuali di Servizio sono:

    - **TCP/IP** – generato per gestire le connessioni in rete delle varie immagini z/VM;
    - **Operator** – macchina virtuale con comandi privilegiati per l'interazione con il CP a disposizione degli Operatori di Sistema;
    - **MAINT o DirMaint (Directory Maintenance)** - per gestione delle "directory" di macchine virtuali;
    - **GCS** - Gestore di Risorse e controllore degli accessi simultanei sui files;
    - **VMRDM** - Performance Toolkit;
    - **RSCS** - Remote Spooling Communication Subsystem.

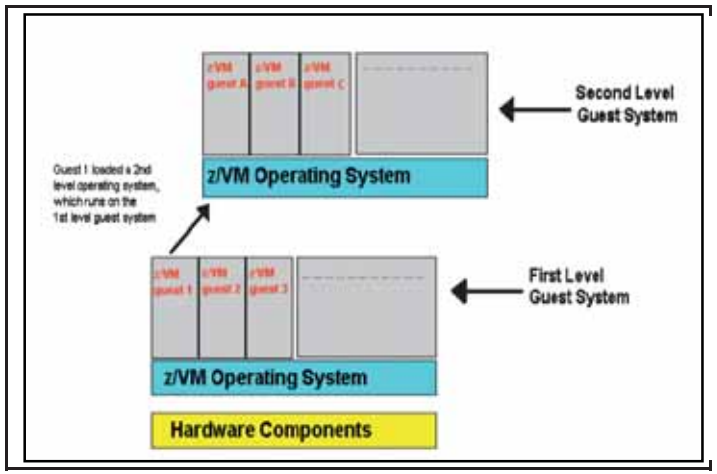


Figura 4 z/VM di secondo livello

3.1.2 Componenti di Base

I componenti di base di z/VM sono il Control Program (CP) e il CMS (Conversational Monitor System).

3.1.2.1 Control Program (CP)

Il Control Program (CP) è un sistema operativo che sta alla base di z/VM. Esso è responsabile della virtualizzazione dell'hardware reale del Sistema Centrale e permette

a molte macchine virtuali di condividere risorse hardware contemporaneamente. Il CP si occupa anche della creazione delle Macchine Virtuali, garantendone la sicurezza, l'integrità e l'isolamento. Il CP però non è un sistema operativo "classico", nel senso che non ha funzioni di gestione dei file e non fornisce un metodo conveniente di caricamento ed esecuzione di programmi differenti e non consente agli utenti di eseguire task produttivi, Web serving, text editing, o data processing.

La sua principale funzione è quella di "Resource Manager". Le risorse che gestisce sono le macchine virtuali. E' poi compito del sistema ospite, sia esso CMS o zLinux o z/OS o altro, usare le risorse fornite dal CP per eseguire i vari Task produttivi.

Come "Resource Manager" il CP permette ad un insieme di Macchine Virtuali di eseguire vari sistemi operativi in un'unica partizione logica (LPAR). Quando un il sistema operativo ospite sarà in IDLE un altro sistema può usare quelle risorse hardware.

La Figura 5 illustra un sistema reale di 5 CPU e 16 GB di RAM in cui le macchine virtuali ospiti hanno tutte insieme un totale di 12 CPU virtuali e 22 GB di RAM.

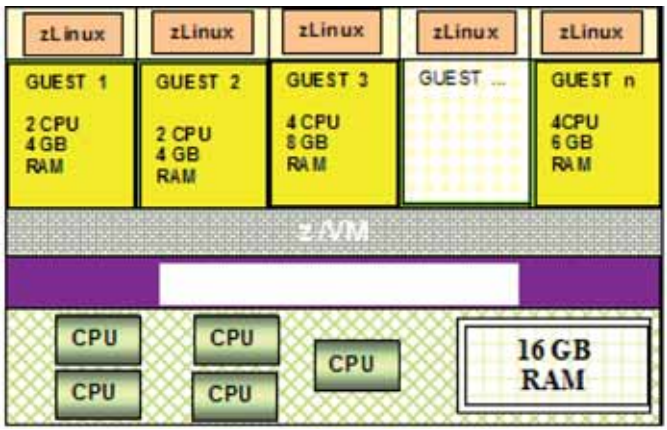


Figura 5 Esempio di ambiente z/VM

Il CP è dotato di un interprete di Comandi accessibile da:

- Interfaccia Line-mode (semplice)
- API programmabile
- Un comando speciale

Questi comandi sono raggruppati in classi di privilegi a seconda dell'utilizzatore (ad esempio System Operator, System Analyst, System Programmer, ecc.). Il CP è anche fornito di:

- Spazio su disco (DASD) riservato per paging spool e dischi temporanei (TDISK)

- Funzioni estese di monitoring, debugging e tuning
- Funzione di error recovery

z/VM fornisce un supporto ai dispositivi di I/O con vari gradi di coinvolgimento del Control Program (CP):

- *Device dedicati o Attached* - Gli ospiti hanno uso esclusivo del device reale e il coinvolgimento del CP è minimo;
- *Device Virtualizzati* - CP presenta un'immagine di un device reale a più macchine virtuali o esso suddivide un device reale affinché una sua porzione appaia reale ad un sistema ospite (per esempio DASD, cripto devices, minidisk). In questo caso il CP è più coinvolto ed inoltre offre anche dei servizi aggiuntivi come ad esempio il minidisk caching;
- *Device emulati/simulati* - CP emula un pezzo di device reale al sistema e ai suoi ospiti come un tipo diverso di hardware oppure il CP simula completamente un device per un ospite senza l'esistenza di quell'hardware reale. Sono esempi di device simulati i Virtual Disk (VDISK), i LANs ospiti, i Virtual Unit Record devices (reader, punch, printer, virtual console). Il coinvolgimento del CP in questo caso è molto alto.

I comandi CP

La via principale per comunicare con il CP è una interfaccia a linea di comandi dalla Console. z/VM usa i comandi CP per configurare e manipolare le risorse, in particolare per controllare processori, dispositivi di I/O, e inoltre per controllare la configurazione delle Macchine Virtuali e il loro ambiente di esecuzione. I Comandi CP hanno parametri posizionali. I nomi dei comandi sono Alfanumerici e minori di 12 caratteri, caratterizzati da verbi che descrivono la funzione; gli operandi sono keywords e simboli minori di 8 caratteri. Essi non sono sempre presenti nei comandi.

Figura 6 Esempio di comandi CP (QUERY VIRTUAL ALL)

Le macchine virtuali comunicano con il CP in una delle due seguenti maniere:

- Un utente (o un tool di automazione al suo posto) può sottomettere comandi CP dalla console virtuale della macchina ospite;
- I programmi che girano nella macchina virtuale possono comunicare direttamente col CP tramite un'istruzione chiamata DIAGNOSE, i cui parametri forniscono al CP tutti i dettagli per ottenere input e fornire una risposta.

L'insieme dei comandi CP e delle numerose funzioni della DIAGNOSE sono suddivisi in gruppi funzionali chiamati classi di privilegi. L'insieme dei comandi e funzioni dell'utente a disposizione su ogni macchina virtuale, come fare un IPL, connettersi ai Minidisk, ecc. è confinato ad un'unica classe di privilegi detta classe G. Ulteriori classi sono specializzate per i System administrator e per gli Operatori.

3.1.2.2 Il CMS

Il Conversational Monitor System (CMS) è un sistema operativo mono-utente virtuale. Esso, infatti, può operare solamente all'interno di una macchina virtuale creata dal CP. Il CMS è utilizzato per facilitare l'amministrazione delle machine virtuali fornendo all'utente un ambiente con una operatività superiore a quella del CP. Il CMS può eseguire una grande varietà di compiti come scrivere, testare e correggere applicazioni sviluppate nel CMS o su Sistemi ospiti, creare ed editare file di dati, condividere dati tra sistemi operativi ospiti e comunicare con vari utenti di sistema.

Una caratteristica interessante è quella che vede la possibilità del CMS di operare da caricatore di un altro sistema operativo (zLinux, z/OS, ecc.) dal suo interno. In effetti, fatto un boot o IPL di un sistema operativo, il CMS non è più visibile ed il controllo passa completamente al sistema operativo caricato. Ogni macchina CMS è definita in System Directory (IPL CMS).

Il CMS è dotato di:

- Un suo file system che generalmente non può essere letto o scritto da altri sistemi operativi;
- Ambiente di sviluppo e operativo per applicazioni, compilatori, ecc.;
- Funzioni caratteristiche:
    - Pipelines - ovvero dei comandi che consentono a più programmi (stages) di passarsi dei records. Le sorgenti dei dati sono varie (comandi, xedit, dischi, storage ecc.)
    - REXX – un potente linguaggio procedurale di programmazione e script
    - XEDIT – un editor molto versatile
    - Interfacce line-mode, full screen e GUI (XPG4)
- Comandi CMS per le sue operazioni.

I files all'interno del file system CMS sono denominati usando un identificatore (file ID) composto da tre campi:

- File name (FN) definito con lettere a A a Z che identificano i minidisk o le directory dove risiede il file;

- File type (FT) è una estensione del nome che ne descrive le caratteristiche (es di filetipe sono TEXT, COBOL , LISTING ,..);
- File mode (FM) o Directory name (dirname) definito con un numero da 0 a 6 che è assegnato quando il file è creato o rinominato (default=1). E' usato per identificare ed operare su un subset di files.

I comandi CMS

La macchina Virtuale CMS è guidata attraverso un considerevole insieme di comandi da una console, come per il CP. Il linguaggio di comandi CMS consente di creare, modificare, fare debug ed in generale maneggiare files. Inoltre i comandi CMS consentono di leggere schede dal card reader virtuale, perforare schede su un card punch virtuale e stampare records su una stampante virtuale. Ci sono anche comandi che aiutano a maneggiare i dischi virtuali, le directory nei file pools e i files sui minidisk.

Un comando CMS consiste in nome di comando, seguito di solito da uno o più operandi posizionali e in molti casi da un lista di opzioni. Il separatore comune in tutti i comandi è il carattere "blank":

Nome comando operandi (opzioni….

Un esempio di comando CMS è il seguente:

COPY GIORNO1 TXT A ARCHIVIO DATA D

che permette di fare una copia del file GIORNO1 TXT sul disco A memorizzandolo sul disco D come ARCHIVIO DATA.

File di configurazione

La maggior parte delle informazioni sulla configurazione del sistema sono contenute in due file ovvero il System Configuration file (SYSTEM CONFIG) e lo z/VM User Directory System file (USER DIRECT).

System Configuration file

Il System Configuration file (SYSTEM CONFIG on MAINT minidisk CFn) definisce le caratteristiche di base del sistema e consente di definire le opzioni principali quali:

- I dispositivi che il CP mette online all'IPL
- I dischi volumi ad uso esclusivo del CP (CP_OWNED)
- I dischi destinati alle macchine virtuali (USER_VOLUME_LIST)
- Il time zone che il CP seleziona tra quelli disponibili

- Le macchine virtuali che vengono attivate automaticamente (AUTOLOG)
- La locazione delle aree di Checkpoint e di ripartenza veloce (warm start del sistema)

Il file SYSTEM CONFIG contiene definizioni che identificano i volumi che sono per uso specifico del CP denominati CP_OWNED. Tali volumi contengono oltre alle aree di spool, di paging, di checkpoint e warmstart anche l'area dove è caricata la SYSTEM CONFIG che viene letta dal CP in fase di inizializzazione di z/VM. Tali volumi devono avere una formattazione/allocazione a blocchi di 4 Kbytes con una "Byte Map" di allocazione sul cilindro (0) del disco. La formattazione e allocazione dei volumi CP_OWNED è effettuata con il programma di utilità CPFMTXA. La Byte Map è costituita da una stringa di Bytes, uno per ogni cilindro di disco. Ogni byte definisce se quel cilindro può essere usato per formattare ed allocare volumi CP_OWNED. La figura 7 illustra le varie aree allocate nei dischi CP_OWNED.

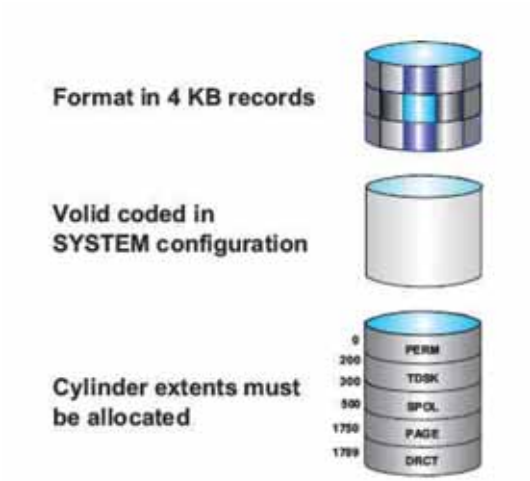


Figura 7 I Dischi CP_OWNED

DRCT Area allocata per il file USER DIRECT
SPOL Area allocata per il files di SPOOL (System Spool Space)
PAGE Area allocata per il file di PAGE
TDISK Area allocata per per il file temporanei (tdisks)
PARM Area allocata per i minidisks di PARM
PERM Area allocata come permanente (a disposizione del CP)

Nella PARM Area sono allocati tre minidischi che sono letti dal CP al tempo di IPL e contengono i parametri di inizializzazione di z/VM. Nel minidisco CF1 è memorizzata la SYSTEM CONFIG.

z/VM User Directory

La z/VM User Directory (USER DIRECT) è un file che è usato per gestire le definizioni di ogni utente. Tenere traccia di queste definizioni può rapidamente diventare complesso all'aumentare del numero di macchine virtuali. Per tale motivo per gestire le user directories in genere viene usato un prodotto denominato IBM Directory Maintenance Facility (DIRMAINT).

La z/VM User Directory descrive al CP la configurazione e le caratteristiche operative di ogni macchina virtuale. La User Directory contiene i nomi e le caratteristiche iniziali di tutte le macchine virtuali attivabili sotto uno z/VM. Questo file può essere editato usando l'editor XEDIT.

Essa è contenuta in forma testuale nel file system del CP. Per ciascuna Macchina Virtuale sono indicate:

- Nome della macchina (un massimo di 8 Caratteri Alfanumerici);
- Password (se non presenti altri prodotti di Security);
- Architettura Hardware della macchina ospite (24, 31, 64 Bit);
- Numero di CPU Virtuali;
- Memoria Virtuale Minima e Massima;
- Minidischi;
- Altri Devices come consoles, spool devices (reader, punch, printer), special devices;
- Opzioni di priorità;
- Il Disco di bootstrap (IPL).

La Directory è composta da una serie di definizioni delle Macchine Virtuali. In figura 8 abbiamo un esempio di definizione di Macchina Virtuale in USER DIRECT.

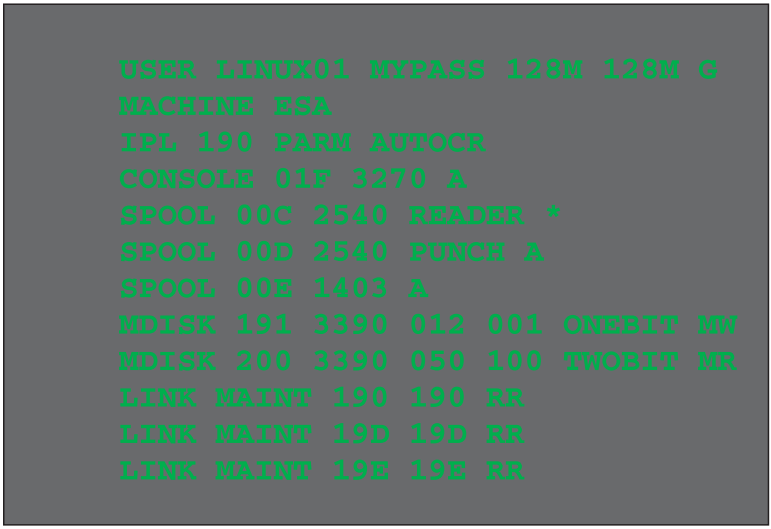


Figura 8 Esempio di definizione in z/VM User Directory

Per poter essere utilizzata dal CP la z/VM User Directory deve essere "compilata" con il programma di utilità DIRECTXA. Una volta compilata da DIRECTXA il file ottenuto da USER DIRECT viene posto nell'area allocata come DRCT in uno dei volumi CP_OWNED (SYSRES).

Il formato del comando è **DIRECTXA fn ft** dove **fn** è il file name della directory (USER è il DEFAULT fn) e **ft** è il file type della directory (DIRECT è il DEFAULT ft).

Le macchine virtuali definite nella USER DIRECT sono "dormienti" fino a che un utente si collega ad esse (LOGON). Esse possono anche essere attivate automaticamente alla partenza di z/VM tramite il Parametro AUTOLOGON nella definizione della Macchina Virtuale in USER DIRECT.

Accesso ai dischi tramite comando di LINK

Il CP fornisce una via per accedere ai dischi "posseduti" da un altro utente del sistema. Ciò avviene con il comando di LINK. Fare il LINK ad un disco è molto utile per condividere i files con un altro ospite del sistema.

Ad esempio tutti gli z/VM networking tools (come telnet, netstat e ftp) sono locati di solito sul disco 592 della macchina TCPMAINT. Per avere accesso al disco 592 della macchina TCPMAINT si usa il comando LINK.

Il comando LINK ha quattro parametri di base:

- il nome della macchina virtuale che possiede il disco richiesto
- il numero di device virtuale di quel disco (relativo alla macchina virtuale che lo possiede)
- il numero di device virtuale con cui la macchina richiedente definisce il disco "linkato"
- un codice di due caratteri per descrivere il livello di accesso Read/ Write al disco.

Un esempio di questo comando è:

LINK MAINT 190 194 RR

in cui la macchina da cui si da il comando richiede un accesso in read only del minidisco 190 della macchina virtuale MAINT. La macchina richiedente al suo interno accederà questo disco con l'indirizzo 194.

La sessione z/VM

Quando un utente entra nel sistema (ovvero fa "logon") il CP crea una macchina virtuale per quell'utente e questa costituirà l'ambiente operativo dove verrà caricato il sistema operativo ospite (CMS, z/OS, ecc.). Questo verrà caricato con un comando di IPL (Initial Program Load) direttamente dal disco dove si trova il Kernel (o Nucleo) del sistema operativo ospite. E' possibile anche caricare un sistema operativo direttamente da una copia di un kernel di sistema operativo salvata precedentemente in memoria utilizzando la funzione "Named Saved Segment (NSS)". Poi in tale ambiente vengono caricate le varie applicazioni.

I modi di esecuzione del CP

Ad ogni istante una macchina virtuale si trova in uno dei due seguenti stati di esecuzione:

- GUEST MODE - ovvero eseguendo istruzioni del sistema operativo ospite
- CP MODE - ovvero non eseguendo istruzioni del sistema operativo ospite

Quando un sistema operativo sta eseguendo in GUEST MODE esso controlla i device e le risorse della macchina virtuale. Tutti i comandi dati alla macchina virtuale saranno interpretati dal sistema operativo ospite e non dal CP.

Però quando la macchina virtuale non sta eseguendo il suo sistema operativo, è il CP ad avere il controllo. Processori e device virtuali della macchina virtuale sono generalmente in IDLE e non consumano risorse reali.

Quando si fa il Logon di una macchina virtuale si è in CP MODE dopo di che quando si fa il boot o IPL del sistema operativo si va in GUEST MODE. Allo Shutdown ovvero alla chiusura ordinata del sistema operativo si ritorna in CP MODE. Tutto ciò accade da un terminale locale o remoto in emulazione 3270.

3.1.3 Dispositivi di I/O in ambiente z/VM

All'interno di z/VM occorre far chiarezza sul tema dei dispositivi reali e virtuali per:

- Unità a Disco
- Unità a Nastro
- Spool devices
- Unità di collegamento alla rete (Network Devices)

3.1.3.1 Unità a Disco

Il Control Program (CP) erige una barriera tra le macchine virtuali e i devices al quale il programma ha accesso. Una funzione primaria del CP è quella di mediare l'accesso ai device reali in vari modi, a seconda che il device debba essere condiviso tra due o più macchine virtuali simultaneamente, come ad esempio un DASD, o se il device debba essere disponibile per l'uso esclusivo di una singola macchina virtuale, come ad esempio una unità a nastro.

Quando una macchina virtuale fa una richiesta di I/O, questa è intercettata dal CP cosicché gli indirizzi sono tradotti nei corrispondenti indirizzi reali. Il CP esamina che la richiesta di I/O non sia impedita da altre I/O prioritarie effettuate da altre macchine virtuali autorizzate e, in caso negativo, inizia l'operazione di I/O al posto della macchina virtuale. Il principale device su cui si memorizzano i dati per z/VM e i dati delle macchine virtuali è il disco (DASD). Esso può essere utilizzato dal sistema operativo z/VM oppure utilizzato dal CP (CP_OWNED) o dagli utenti (USER_VOLUME_LIST) per:

- Checkpoint data, Warmstart data, Paging, spooling data
- Minidisks dei sistemi ospiti (tramite MDISK z/VM System Directory Statement)
- Dischi temporanei dei sistemi ospiti (tramite comando CP DEFINE T3390).

I dischi virtuali (virtual DASD) usati dalle macchine virtuali sono chiamati "minidisk". Essi sono creati dal partizionamento di dischi fisici reali (Real DASD) in intervalli di cilindri che appaiono come dischi al server virtuale. Un minidisk può anche essere un intero disco reale.

Esistono tre tipi di minidisco:

- **Permanente** o residente ovvero memorizzato su una porzione di disco reale che rimane memorizzato indipendentemente dalle sessioni. Esso è assegnato ad una o più macchine virtuali con una definizione in z/VM System Directory;

- **Temporaneo** (TDISK) ovvero minidisco virtuale temporaneo usato per appoggiare dei dati all'interno di una sessione VM (LOGON). Tali dati si perdono alla fine della sessione (LOGOFF). Si possono definire TDISK di varie grandezze senza preoccuparsi della locazione su disco (non dobbiamo perciò preoccuparci della contiguità dei cilindri allocati);

- **Disco virtuale** (VDISK) **in memoria** ovvero un dispositivo di I/O ad alta velocità in grado di compiere le stesse operazioni dei dischi fisici anche se allocati in Memoria Virtuale e quindi paginati sulla memoria reale di z/VM. I Dischi virtuali sono volatili. Se accade un errore grave nel CP o uno shutdown i dischi virtuali sono persi. Esso ha le stesse caratteristiche fisiche del disco fisico (velocità, geometria, ecc.). Mentre è in uso, i blocchi di VDISK sono in memoria centrale. Ciò è molto vantaggioso in termini di performance. Quando non è in uso esso, se necessario, può essere ospitato sui paging dataset. Poiché il dato del VDISK è volatile esso dovrebbe essere usato per file temporanei (ad esempio assembly, VSE lock file, or sort applications). Un VDISK può essere creato tramite comandi CP o all'interno di una definizione di MDISK nella z/VM System Directory. Virtual Disk (VDISK) creati all'interno della definizione MDISK sono condivisibili da più macchine virtuali. Il vantaggio dei VDISK è che l'operazione di I/O è in effetti trasferita dal device fisico per i files residenti al paging I/O con vantaggi di performance. (nel caso ci sia sufficiente memoria reale addirittura non si effettua neanche il paging I/O). L'area di paging VM dedicata ai Virtual disk occupa circa un quarto del totale.

3.1.3.2 Unità a Nastro Virtuale

In maniera del tutto analoga ai dischi abbiamo la possibilità di condividere le unità a nastro fisiche tra varie macchine virtuali utilizzando le tecniche di virtualizzazione. Il nastro virtuale, come quello reale, può essere montato, letto, scritto, riavvolto e scaricato.

3.1.3.3 Spool devices

Nei tempi in cui fu creato il VM la maggior parte delle installazioni comprendevano stampanti, lettori e perforatori di schede reali chiamati Spool Devices. L'uso di tali dispositivi virtuali mantiene la sua fondamentale importanza nella operatività abituale di z/VM anche se i dispositivi reali che li denominano sono obsoleti. I dati coinvolti in input o in output su tali dispositivi sono memorizzati sul System Spool Space. Lo z/VM riesce a emulare questi tre tipi di device per gli utenti:

- **Reader** o RDR cioè lettore di schede - è un dispositivo di input delle macchine virtuali. Per ciascuna macchina virtuale è riservata un'area di Spool detta RDR in cui è possibile accodare file in input. Il formato del file deriva da quello originario delle schede perforate e cioè un insieme di record di 80 caratteri. L'uso principale che viene fatto attualmente del reader è la funzione di mailbox dei files ad esempio con il comando CMS SENDFILE si può trasferire un file da una macchina virtuale ad un'altra indirizzandolo alla coda di RDR della macchina ricevente. Rimane anche la funzione orginaria del Card Reader RDR di area di input per sottomettere i JOB con un linguaggio che viene interpretato dallo spool;
- **Punch** o PCH ovvero perforatore di schede - indica il dispositivo virtuale di output della macchina virtuale. Per ciascuna macchina virtuale è riservata un'area di Spool detta PCH in cui è possibile accodare file in output. Nell'esempio precedente la macchina che esegue il comando CMS SENDFILE

spedisce il file attraverso il PCH che lo indirizza nella coda di RDR della macchina ricevente;

- **Printer** o PTR cioè stampante - ovvero il dispositivo di stampa di un output. Per ciascuna macchina virtuale è riservata un'area di Spool detta PTR in cui è possibile accodare file in output in formato di stampa.

3.1.4 Gestione dispositivi di rete

Le tecniche di virtualizzazione di z/VM consentono implementazioni molto interessanti nell'ambito delle configurazioni di rete. Normalmente la connettività hardware in rete è garantita da una scheda OSA (Open System Adapter). Quindi sia z/VM che le macchine virtuali ospiti si possono presentare in rete utilizzando questo canale (vedi figura 9).

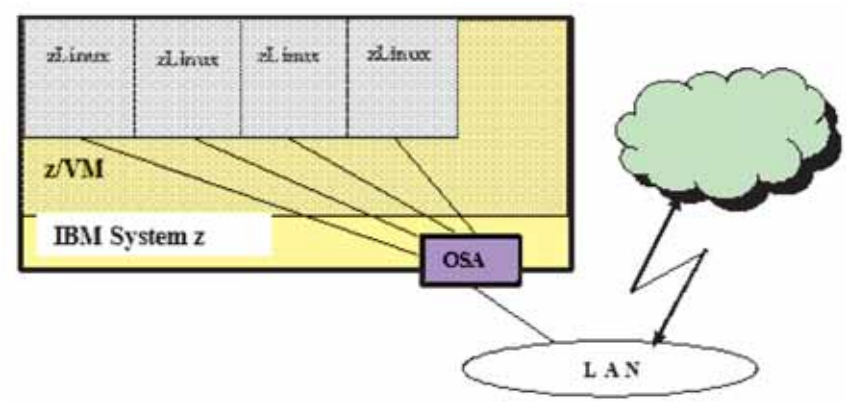


Figura 9 Ambiente z/VM e rete

La comunicazione tra macchine virtuali è fornita da vari dispositivi simulati o da funzione proprie di z/VM. Esistono percorsi di comunicazione tradizionali di z/VM quali InterUser Communication Vehicle (IUCV) o Virtual Channel to Channel. Oltre ad essi, z/VM utilizza la funzione HW QDIO e il QDIO CP subchannel per fornire due tipi di reti virtuali per la comunicazione tra macchine virtuali all'interno di z/VM: z/VM guest LAN e z/VM Virtual Switch.

3.1.4.1 z/VM Guest LAN

z/VM Guest LAN consente di creare delle reti virtuali di sistemi ospiti all'interno del sistema z/VM. Questa funzione permette di creare un numero indefinito di reti virtuali mediante definizioni contenute nel file di configurazione SYSTEM CONFIG o con il comando CP DEFINE LAN. Ogni z/VM Guest LAN è isolata dalle altre a meno che non vi sia un TCP/IP Router che le connetta. Ciò è vero anche nel caso che le z/VM Guest LAN debbano comunicare con reti esterne a z/VM. Il TCP/IP router è una macchina Virtuale CMS specializzata nell'indirizzamento del traffico TCP/IP. Gli Adapter utilizzati da zVM Guest LAN sono di tipo HiperSockets (nel caso di comunicazioni interne al

Sistema Centrale) e OSA Express .Tali Adapter sono all'interno dello Stack TCP/IP delle Network Interface Card Virtuali. Ogni machina virtuale possiede una Network Interface Card Virtuale che è connessa alla Guest LAN. Essa fornisce l'accesso alle z/VM Guest LAN o ai Virtual Switch (vedi figura 10).

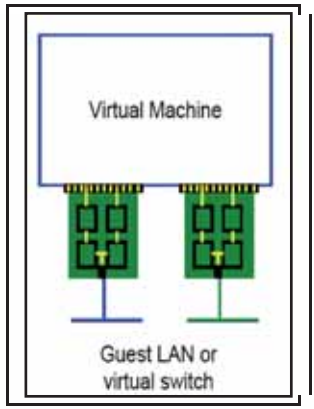


Figura 10 Virtual Network Interface Card (NIC)

Le connessioni dedicate sono più performanti in presenza di un carico di rete intenso o necessarie quando è richiesto un accesso diretto alla rete fisica o quando si ha bisogno di una banda di rete garantita. Le z/VM Guest LAN risultano utili quando si hanno risorse hardware limitate, quando molti nodi sono all'interno dello stesso z/VM o quando occorre isolare un certo carico di rete (ad esempio ambienti di test) dalla rete primaria. Le Guest LAN possono utilizzare i protocolli Ethernet IPv4 e IPv6 (vedi figura 11).

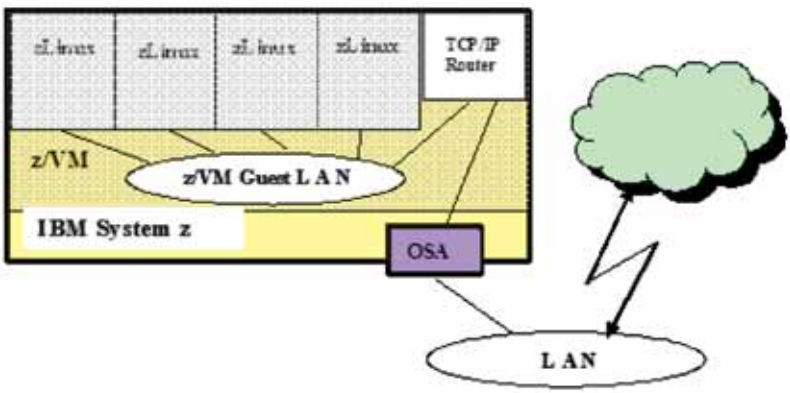


Figura 11 z/VM Guest LAN

3.1.4.2 z/VM Virtual Switch

Le Guest LAN interne a z/VM possono connettersi a reti esterne al sistema che ospita z/VM direttamente via adapter di tipo OSA-Express. In alternativa in ambiente z/VM si possono utilizzare gli z/VM Virtual Switch.
z/VM Virtual Switch connette una LAN esterna e una o più Guest LAN. Simile ad una Guest LAN basata su interfaccia virtuale di rete di tipo OSA-Express, z/VM Virtual Switch opera solamente con adapter di tipo OSA-Express. Esso supporta il trasporto di pacchetti IP e di frame Ethernet. Gli z/VM Virtual Switch non richiedono un router per connettersi alla rete esterna, come illustrato in figura 12.

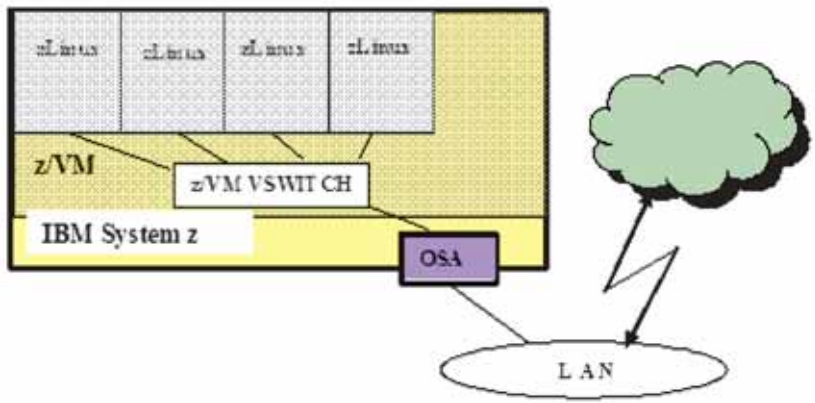


Figura 12 z/VM Virtual Switch

Si può a questo punto richiamare il concetto di Virtual LAN (VLAN) che è complementare alle configurazioni di z/VM Guest LAN e z/VM Virtual Switch. Lo standard IEEE 802.3 definisce una Virtual LAN (VLAN) ovvero la funzione che consente a una rete fisica di essere partizionata in reti logiche separate. Dal punto di vista operativo queste reti sono fisicamente indipendenti (separazione dei domini di collisione).

Esse non vanno confuse con le z/VM Guest LAN sebbene le z/VM Guest LAN e i Virtual Switch possono far parte di una VLAN. Tra i dispositivi simulati che supportano le VLAN c'è il Virtual Switch.

Relativamente al Virtual Switch lo standard IEE 802.1Q regola la gestione dell'assegnazione di utenti z/VM a specifiche VLAN e assicura che essi riceveranno solo i pacchetti appartenenti alle VLAN cui gli utenti sono autorizzati. Nel caso di Guest LAN sarà necessario un router (una macchina virtuale specializzata) che connetta la rete interna ad una subnet esterna differente. Nel caso di Virtual Switch ciò non sarà necessario in quanto la subnet esterna sarà la stessa (con un supporto di transparent bridge – vedi figure 13 e 14).

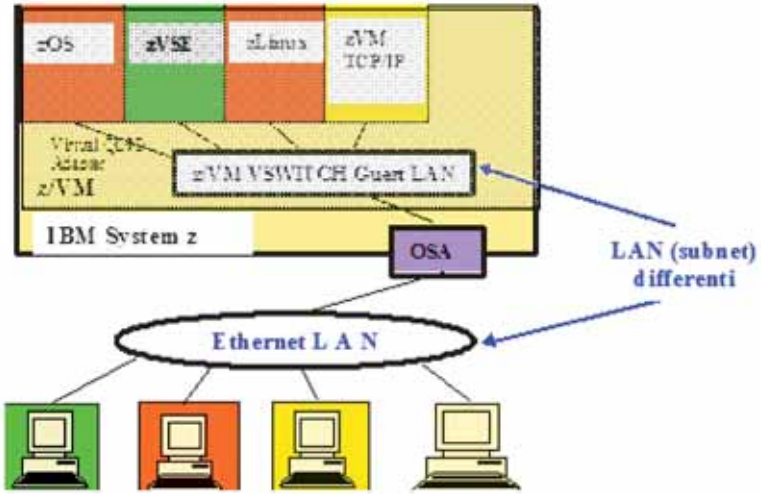


Figura 13 Configurazione Virtual Switch con VLAN

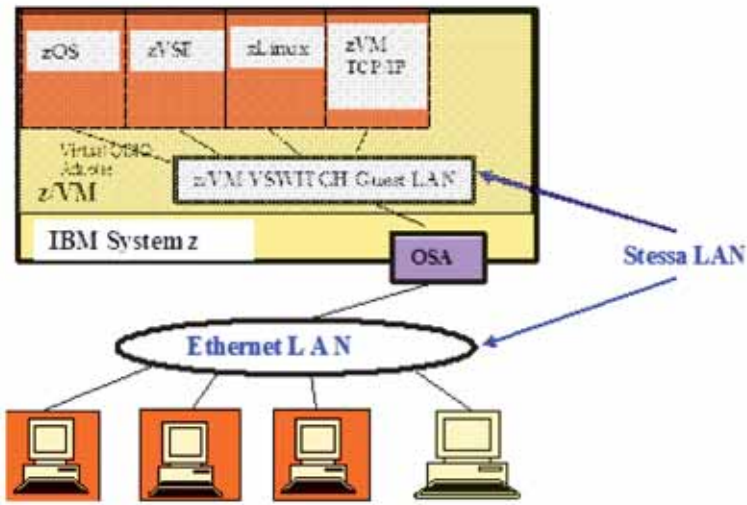


Figura 14 Configurazione Virtual Switch senza VLAN

3.1.5 Virtualizzazione di CPU e Memoria

La virtualizzazione del processore viene affrontata da z/VM con l'obiettivo di dare ad un sistema ospite "la convinzione" di avere il controllo esclusivo del processore. In effetti i processori sono condivisi tra i vari sistemi operativi ospiti.

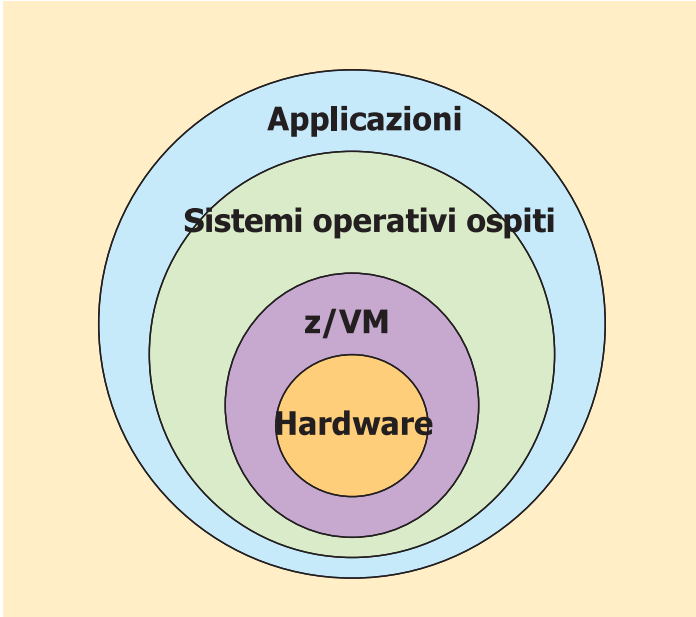


Figura 15 Livelli di Virtualizzazione

Andiamo a considerare dunque i virtual processor ovvero i CP che la Virtual Machine ha a disposizione. Essi possono essere CP *dedicati o condivisi*. Ovvero una macchina virtuale può ottenere la risorsa CP in modo esclusivo (dedicato) nel tempo, oppure può condividere questa risorsa con altre macchine virtuali. Ciò avviene in maniera dinamica nel tempo, all'interno del concetto di time slice (che rimane la base di coesistenza delle macchine virtuali in z/VM). In tal senso anche i CP virtuali dedicati sono utilizzati dinamicamente.

La condivisione di CP può essere assoluta o relativa, vale a dire che i vari livelli di virtualizzazione consentono varie tecniche (per esempio hard e soft capping) che assegnano parti della risorsa CP a seconda delle necessità. I vantaggi della condivisione della risorsa CP possono consentire ad una macchina virtuale l'utilizzo di tutta la potenza elaborativa in mancanza di contesa con una o più macchine virtuali potenzialmente abilitate a quello stesso CP.

Un altro vantaggio legato alla virtualizzazione e alla condivisione delle CPU virtuali è il cosiddetto "over commitment" ovvero la definizione di risorse virtuali che sono sovrabbondanti rispetto a quelle fisiche. Infatti se una macchina non riesce ad utilizzare tutta una sua risorsa, la parte da essa non utilizzata può essere disponibile ad altre risorse.

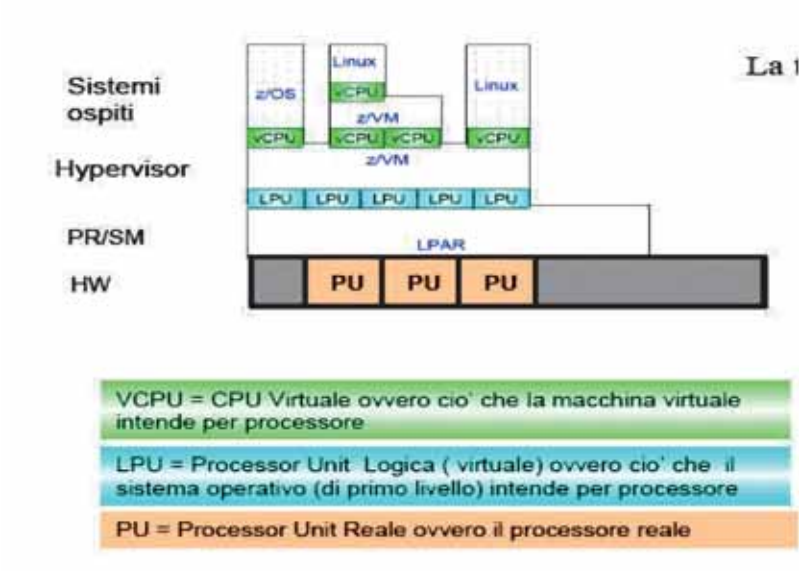


Figura 16 Virtualizzazione della CPU

La gestione della memoria virtuale di z/VM è quella legata alla architettura tipica dei Sistemi Centrali già descritta precedentemente. Tramite una funzione di traduzione degli indirizzi DAT (Dynamic Address Translation) z/VM può creare, gestire e isolare spazi di indirizzamento (address space).

Ogni address space è costituito di un insieme di tabelle (region, segment, page tables) associate che contengono informazioni precise sulle locazioni di memoria reale che vengono usate dai processi. Queste tabelle sono usate dalla DAT per convertire indirizzi di memoria virtuale in indirizzi di memoria reale. Poiché queste tabelle sono matenute dal sistema operativo e non sono accessibili dai processi stessi, non è possibile per loro leggere e/o scrivere in memoria ciò che è usato dal sistema operativo o da un altro processo.

z/VM va oltre e utilizza questa capacità attraverso due livelli di traduzione di indirizzi. Un sistema operativo che opera in una macchina virtuale sotto z/VM costruisce le sue tabelle di traduzione degli indirizzi come di consueto ai fini di isolare e contenere i suoi processi. L'intera memoria reale di questa macchina virtuale, sebbene vista da un sistema operativo ospite come memoria reale, è nei fatti una memoria virtuale definita da un insieme di tabelle gestite dal Control Program (CP) di z/VM.

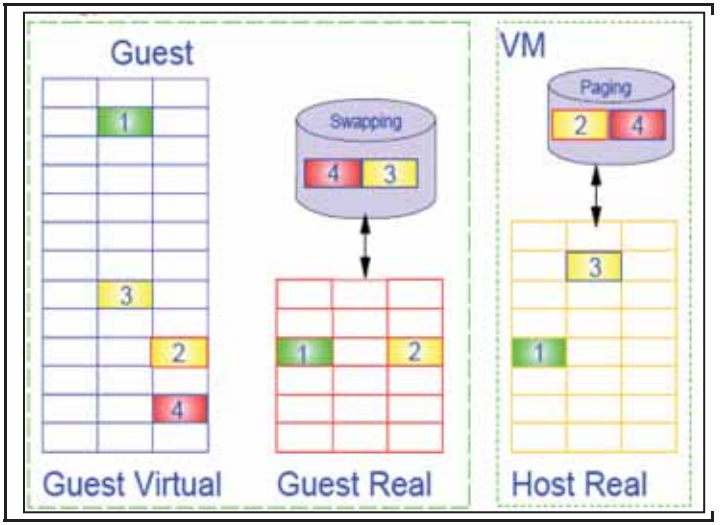


Figura 17 z/VM virtual memory

Esiste una funzione specifica di z/VM che pemette a più macchine virtuali di condividere memoria virtuale. Questa funzione è chiamata Shared Segments e permette di ridurre drasticamente la quantità di pagine duplicate di codice e di dati. Gli Shared Segments possono essere blocchi di memoria read-only o read-write. Ad esempio molte macchine virtuali Linux possono condividere il codice read-only del kernel. Accade che più di una segment table di macchine virtuali punterà alle stesse page frames in memoria reale.

z/VM ha un meccanismo di protezione della memoria per non far alterare senza il suo assenso i segmenti salvati dichiarati in read-only. Esiste anche la possibilità di condividere segmenti salvati in read-write mode. z/VM permette ad una sola macchina virtuale di definire e salvare gli shared-segments.

3.1.6 Scheduling e Dispatching

z/VM gestisce le Macchine Virtuali e assegna loro la risorsa elaborativa utilizzando i componenti del sistema operativo denominati Dispatcher e Scheduler. Le Macchine Virtuali che divengono attive (al Logon) e che concorrono per ottenere la CPU vengono inserite e accodate in tre liste *dormant, eligible e dispatch list*. Quando si fa Logon allo z/VM, il CP crea un Blocco di controllo (control block1) che è un descrittore di un grande numero di informazioni della Macchina Virtuale. Tale blocco è denominato Virtual Machine Descriptor Block (VMDBK).

Lo scheduler si occupa di priorità all'interno delle liste e degli spostamenti dei VMDBK tra queste. Il dispatcher è quello che toglie gli utenti dalla dispatch list, permette loro di eseguire e di ritornare eventualmente nella Eligible list.

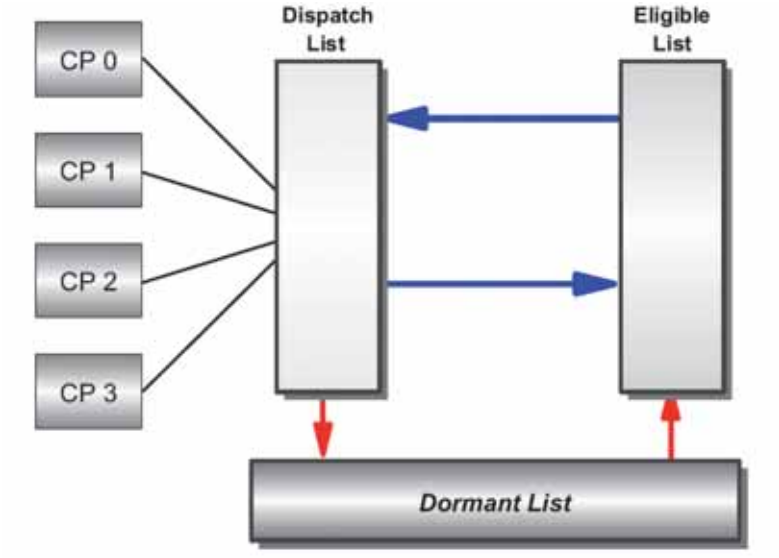


Figura 18 Liste di Dispatcher

Questo blocco sarà puntato dalle varie liste di scheduler/dispatcher che sono mostrate in figura 18. Se la macchina virtuale non sta facendo lavoro produttivo il CP la pone nella dormant list che indica che essa è "dormiente". Se per tale macchina si inizia a lavorare, ad esempio premendo un Enter Key dalla console, essa è posta in una lista che indica che essa riceverà delle risorse di sistema (eligible list).

Prima che sia data qualsiasi risorsa, però, la macchina virtuale è esaminata dallo scheduler. Lo scheduler guarda nel VMDBK della macchina virtuale per vedere quante risorse elaborative (Processori, memoria e paging) ha usato in passato.

Se si è alla fase di Logon non ci saranno informazioni accumulate nel VMDBK perciò lo scheduler assumerà o meglio caratterizzerà la macchina virtuale come se fosse un Utente interattivo anche detto utente di Classe 1.
Un utente di Classe 1 è normalmente un utente interattivo CMS o un sistema operativo ospite che esegue transazioni veloci.
Quando lo scheduler determina che ci sono risorse sufficienti a soddisfare la richiesta della Macchina Virtuale senza porre altri utenti in difficoltà, questa viene spostata in dispatch list dove attenderà il suo turno per accedere alla risorsa CPU per un certo intervallo di tempo (time slice).
Quando arriva il suo turno, la Macchina Virtuale verrà fatta eseguire su un processore disponibile. Il tempo riservato a tale macchina, conosciuto come dispatcher minor time slice, ad un certo punto si esaurirà.
Se la Macchina Virtuale in oggetto ha ancora lavoro da svolgere, essa viene mantenuta in dispatch list e al suo turno otterrà un altro time slice fino a che o avrà finito di lavorare (nel qual caso sara spostata in dormant list) o lo scheduler di nuovo stimerà la richiesta di risorse prima di rischedulare tale macchina virtuale.

Nel caso in cui la macchina ritorni molte volte in dispatch list, lo scheduler può decidere che essa non si caratterizza più come un utente interattivo e che ha bisogno di un quantità maggiore di risorsa elaborativa. Tale Macchina Virtuale viene allora innalzata a utente di Classe 2 che gli permetterà di rimanere in dispatch list per intervalli più lunghi che consentano di finire il lavoro .

Questo ha un vantaggio ulteriore perché riduce il numero di decisioni che il dispatcher deve prendere per gestire gli utenti della elegible list. Un utente di Classe 2 può essere tipicamente un programma di compilazione di un utente CMS o un Web Server zLinux.

Dopo che la Macchina Virtuale è ospitata in dispatch list come utente di Classe 2 molte volte e non ha ancora finito il suo lavoro, lo scheduler la innalzerà alla Classe 3 che le assegnerà un intervallo più lungo per finire il lavoro. Di solito un utente di Classe 3 sarà un sistema operativo (ospite) di produzione.

Ci sono degli svantaggi a entrare in Classi di dispatching più alte. Per esempio se le risorse di sistema (come paging o Memoria) diventano critiche, allora lo scheduler metterà gli utenti delle Classi più alte in eligible list e lascerà in dispatch list quelli delle classi più basse.

Però c'è un tipo di utente conosciuto come utente di Classe 0 che non sarà mai in eligible list (questo avrà l'opzione OPTION QUICKDSP nella sua entrata in USER DIRECT). Tipicamente questo tipo di utente è un utente speciale o un server di primaria importanza. La figura 19 descrive le varie classi di dispatch list.

Classe di utente	Spiegazione
Classe 0	Classe di utenti che furono aggiunti in dispatch list senza ritardi in elegible list, senza tener conto della lunghezza del lavoro da svolgere
Classe 1	Classe di utenti che hanno appena iniziato una transazione e perciò si assume che stiano processando transazioni corte
Classe 2	Classe di utenti che non hanno terminato la transazione corrente nella loro prima pemanenza in dispatch list e perciò si assume che stiamo processando transazioni medie
Classe 3	Classe di utenti che non hanno terminato la transazione corrente nella loro seconda pemanenza in dispatch list e perciò si assume che stiamo processando transazioni di lunghe

Figura 19 Classi di dispatching

3.1.7 Start Interpretive Execution (SIE)

La virtualizzazione in z/VM è supportata da una funzione chiamata Interpretive Execution che consente una interazione del sistema operativo con l'hardware sottostante tale da consentire al contempo performance e integrità. La z/Architecture fornisce la base della capacità del CP di mantenere l'integrità del sistema. Il CP si avvale di una funzione denominata Interpretive Execution Facility che permette ad un

flusso di istruzioni di una Macchina Virtuale di essere eseguite su di un processore usando una singola istruzione chiamata Start Interpretive Execution (SIE) eseguita dal CP. Una Macchina Virtuale è attivata e le sue operazioni iniziano quando il dispatcher del CP esegue tale istruzione.

La SIE ha come unico operando un puntamento ad un blocco di controllo in memoria chiamato State Description il quale specifica lo stato e l'architettura della Macchina Virtuale ove un programma chiamato Programma Ospite (guest program) deve essere eseguito (il programma ospitante che esegue la SIE è chiamato host program).

Il blocco di controllo State Description contiene anche informazioni come PSW (Program Status Word), registri, CPU timer, TOD clock, Interrupts pendenti, ecc. del programma da eseguire.

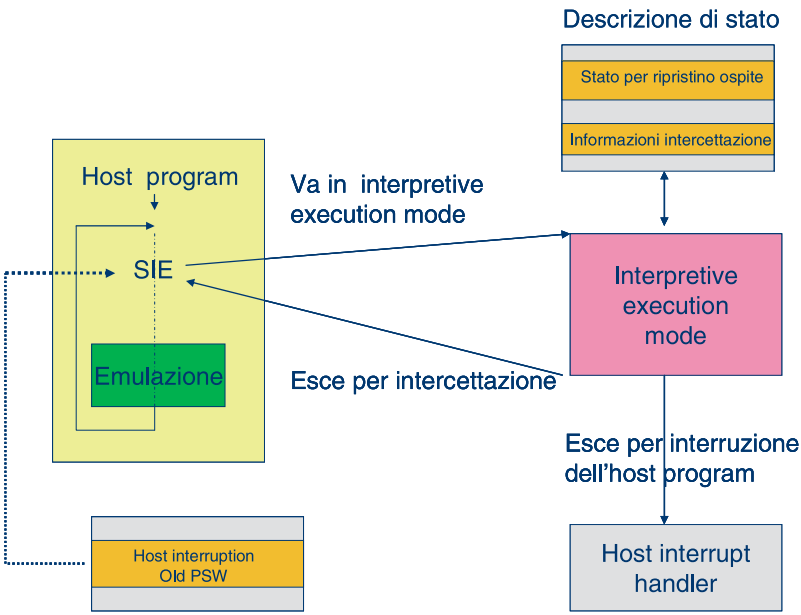


Figura 20 Start Interpretive Mode (SIE)

La maggior parte delle funzioni del Programma Ospite sono eseguite dalla Macchina Virtuale. Quando avviene una condizione che richiede l'intervento del CP si ha una intercettazione. La macchina lascia l'interpretive mode e trasferisce il controllo al CP.

Questa include alcune condizioni come l'esecuzione di alcune istruzioni privilegiate, in particolare istruzioni di I/O, gestione di certe interruzioni dei sistemi ospiti e del wait state della PSW ospite. La macchina lascia l'interpretive mode anche quando un external interrrupt o I/O interrupt causa l'interruzione del programma ospitante.

Quando la macchina lascia l'interpretive mode, il control block che descrive lo stato è aggiornato per riflettere lo stato corrente del Programma Ospite cosicché in seguito l'esecuzione del Programma Ospite può essere ripristinata senza ulteriori cambiamenti alla descrizione dello stato.

Quando accade una intercettazione, l'informazione è inserita nella descrizione dello stato per facilitare l'identificazione e l'analisi delle condizioni dell'ospite da parte del programma ospitante.

Quindi le macchine virtuali possono operare in privileged mode ovvero eseguire istruzioni privilegiate senza che vengano intercettate dal CP, ad eccezione di istruzioni di I/O che vengono simulate/emulate dal CP stesso.

3.2 Linux nei Sistemi Centrali IBM

Linux è un sistema operativo il cui kernel è mantenuto da Linus Torvalds e il primo rilascio risale al 1991. Il codice del kernel è Open Source e nel corso degli anni Linux è evoluto diventando un sistema operativo ampiamente supportato da un crescente parco software. Oggi le distribuzioni Linux affiancano al kernel una varietà di tool e programmi di utilità forniti dalla comunità Open Source, ad esempio dal progetto GNU, creando un sistema completo.

Linux è caratterizzato da una forte indipendenza dalla piattaforma e viene eseguito su molte architetture, incluse Intel, Alpha o Sparc. Linux on System z è un port di Linux a 64 bit nell'architettura S/390; è un Linux "puro" dal punto di vista utente e supporta l'architettura dei processori e i dispositivi che sono specifici degli ambienti mainframe.

IBM ha introdotto Linux sui Sistemi Centrali nel 2000. Si è trattato inizialmente di un esercizio accademico di una Università americana (Marist College) dove alcuni ricercatori hanno modificato il kernel per supportare l'hardware dei grandi Sistemi scrivendo, ad esempio, i driver per gestire le interfacce di I/O, i dischi, i nastri magnetici e le schede di rete. Con il contributo di altri laboratori IBM Europei e della comunità Open Source l'esercizio è diventato presto un codice affidabile incluso nelle più comuni distribuzioni. Linux on System z è un'implementazione ASCII nativa, che pertanto entra a fare parte delle opzioni possibili per Linux al pari delle altre. IBM rilascia regolarmente sul sito di Developerworks le patch da applicare ai sorgenti del kernel per supportare le funzioni dai nuovi hardware.

L'interesse del mercato per questa implementazione è dovuto alla capacità dei mainframe di ospitare un numero elevato di sistemi Linux (in cui generalmente viene ospitata un'applicazione o componente strutturale di un'applicazione, ad esempio il database) attraverso le tecniche di Partizionamento e Virtualizzazione. Tali tecniche permettono un utilizzo efficiente della capacità installata, riducendo ad esempio in maniera sensibile i costi legati alle licenze software. Per i clienti che hanno significativi carichi tradizionali (CICS, IMS, etc.) Linux on System z rappresenta un importante abilitatore tecnologico per i progetti di modernizzazione in quanto permette un'efficace "esposizione" di tali asset verso gli altri sistemi (ad esempio in ottica SOA - Service Oriented Architecture).

3.2.0 Utilizzi di Linux in ambienti enterprise

In ambito aziendale, gli usi più frequenti di Linux, illustrati in figura 1, sono:

- realizzare applicazioni aziendali distribuite;
- realizzare soluzioni infrastrutturali;
- consolidare i workload aziendali;
- creare cluster ad alta capacità di calcolo.

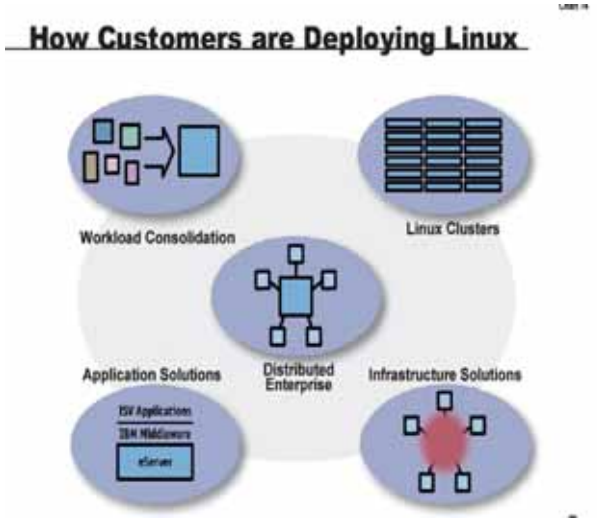


Figura 1 Alcuni utilizzi di Linux

In un'infrastruttura informatica aziendale i server Linux possono svolgere pressoché qualunque funzione, tra cui:

- Web Application Server;
- Database server;
- File Server e Print Server;
- Content/Caching Server;
- Security Server.

3.2.1 Le applicazioni su zLinux e il processo di porting

Il numero delle applicazioni disponibili su zLinux cresce costantemente e molti dei prodotti middleware IBM sono supportati in tale ambiente. Anche molti software sviluppati da importanti ISV, tra cui Oracle e SAP, sono forniti per zLinux.

Il processo di porting consiste principalmente nell'adattare il codice sorgente di un programma applicativo dall'ambiente operativo in cui è stato sviluppato ad un altro ambiente operativo. Tale attività richiede normalmente un considerevole sforzo e competenze specialistiche, in quanto è necessario conoscere approfonditamente le piattaforme di partenza e di arrivo: ciò per rimappare le funzioni utilizzate dal codice su un sistema verso funzioni equivalenti presenti sul sistema di arrivo (ad esempio da sistemi Unix a Linux).

Il codice applicativo scritto per Linux (ad esempio in C/C++) è caratterizzato invece da una maggiore portabilità tra le architetture supportate dal sistema operativo. In tal senso è spesso possibile utilizzare su zLinux un programma disponibile su altre varianti di Linux semplicemente ricompilandone il sorgente. Fanno eccezione gli applicativi il cui codice dipende da particolari caratteristiche di una architettura, quali la presenza di particolari device o la endianness propria dell'architettura.

Generalmente, le applicazioni basate su Java possono essere portate facilmente, a patto che sia presente sull'ambiente target una Java virtual machine del livello richiesto.

Ci sono varie considerazioni che devono essere affrontate durante il porting di una applicazione per Linux su System z:

- l'applicazione che contiene codice specifico dell'architettura e/o specifico dell'assembler deve essere reimplementata per la z/Architecture;
- il middleware, le librerie e i database usati dall'applicazione devono essere disponibili per zLinux;
- il sistema z è di tipo big endian; qualsiasi codice che processa dati orientati ai byte che hanno origine su un sistema little endian potrebbero richiedere che alcuni campi siano invertiti. Potrebbe essere necessario rigenerare i dati o, se questo non è possibile (per esempio nel caso di file condivisi), potrebbe essere necessario ritoccare l'applicazione in modo che processi i dati in formato little endian;
- per una migrazione che coinvolge le applicazioni Web, la maggior parte dei contenuti può essere direttamente trasportata senza alcun cambiamento su qualsiasi Web Server che gira su zLinux;
- se si utilizzano applicazioni web basate su Windows che adottano DLL implementate ad hoc, esse devono essere riscritte usando linguaggi di scripting compatibili (ad esempio Perl) e poi portati su Linux per System z;
- per applicazioni che usano Oracle e DB2 come backend, i database possono essere direttamente trasportati su Linux per System z utilizzando gli strumenti messi a disposizione dal middleware. Nel caso si utilizzino altri database (ad esempio SQL Server), è necessaria una migrazione verso un RDBMS supportato.

Maggiori dettagli si possono trovare ad esempio nell'articolo Linux for S/390 and zSeries porting hints and tips[1].

3.2.2 Linux sul Mainframe – Modalità Operative e Risorse Gestite

Linux su Mainframe può essere utilizzato nelle seguenti modalità:

- in una partizione Logica (LPAR) come Sistema Operativo Primario; tutte le risorse dell'Hardware o della Partizione Logica sono viste e possedute dal sistema operativo Linux;

[1] http://www-128.ibm.com/developerworks/eserver/articles/linux_s390/index.html

- in una macchina virtuale definita e controllata da z/VM. z/VM presenta a Linux le risorse Virtuali come se fossero Reali; generalmente le risorse hardware sono possedute e gestite da z/VM.

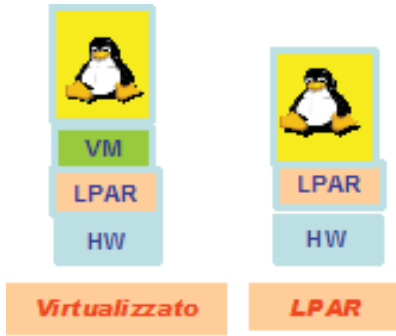


Figura 2 Modalità operative di Linux su mainframe

Linux fornisce metodi di accesso per:

- Dischi (DASD) tradizionali (formato ECKD);
- Dischi e nastri SCSI;
- Schede Ethernet;
- TCP/IP Interno (Hipersocket).

Inoltre esistono driver addizionali per:

- Nastri tradizionali;
- Schede Token Ring;
- Protocolli di comunicazione z/VM (CTCA / IUCV);
- Terminali 3270.

I dischi (DASD) possono essere acceduti sia con il protocollo di I/O del mainframe (ad esempio FICON) sia con il protocollo SCSI su fibra (FCP). Attualmente il protocollo FCP fornisce, per alcuni workload, prestazioni migliori di quelle ottenibili con il protocollo FICON. Tuttavia con il protocollo mainframe si ottengono generalmente migliori livelli di affidabilità. Nel caso FCP il multipathing, ad esempio, è gestito a livello di sistema operativo, mentre nel caso FICON è integrato nativamente nell'architettura a canali (livello hardware). In ambiente z/VM un disco fisico (DASD) può essere suddiviso in più dischi logici (minidischi) assegnabili alle diverse immagini virtuali Linux. Ciascuna macchina virtuale potrà utilizzare il minidisco come se fosse un normale DASD.

Generalmente le risorse dedicate a Linux vengono gestite su mainframe con uno speciale protocollo di tipo DMA – Direct Memory Access detto Queued Direct I/O (QDIO). Grazie a questo protocollo, la path length delle istruzioni di I/O viene

sostanzialmente ridotta rispetto all'architettura a canali. Ad esempio, le schede OSA Express (interfacce di rete) possono essere configurate per usare il QDIO (vedi figura 3).

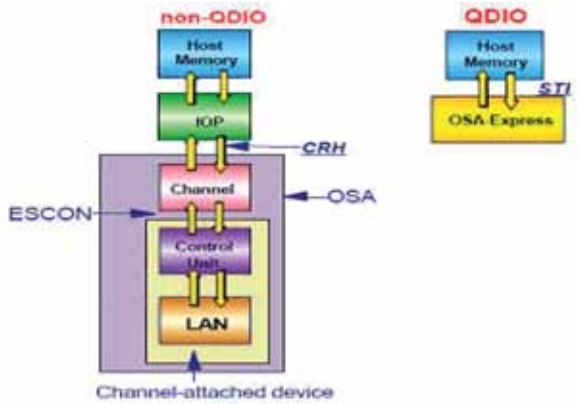


Figura 3 Modalità di accesso ai device

I dettagli relativi ai driver specifici dell'ambiente Linux su mainframe sono reperibili sul sito:
http://www.ibm.com/developerworks/linux/linux390/october2005_documentation.html

3.2.3 Uso di processori specializzati (IFL)

Sui sistemi z è possibile utilizzare i processori specializzati IFL (Integrated Facility for Linux) dedicati all'esecuzione esclusiva di carichi di lavoro zLinux che non incrementano la capacità installata per i carichi tradizionali. Le partizioni logiche (LPAR) sulle quali è attivato Linux o z/VM con soli sistemi ospite di tipo Linux possono utilizzare processori standard (CPU) o processori specializzati di tipo IFL sia dedicati che condivisi. La Figura 4 mostra una possibile configurazione con processori CP e IFL assegnati a partizioni con sistemi ospite di tipo Linux.

A paritire dal System z10 è possibile attivare con z/VM 5.4 o superiore partizioni logiche in modalità z/VM (z/VM LPAR mode). In questa modalità una partizione z/VM può gestire una combinazione di guest che utilizzino CP, zIIP e zAAP (quali z/OS) e IFL (zLinux). Questa funzione permette di implementare un'intera soluzione architetturale z/OS + zLinux in un'unica partizione z/VM, ad esempio per testarne il funzionamento prima di un passaggio in produzione.

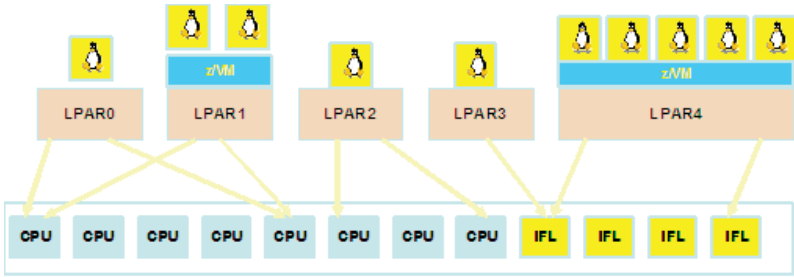


Figura 4 zLinux e processori mainframe

3.2.4 Networking

Linux on System z è caratterizzato da diverse opzioni di networking. E' possibile raggruppare le opzioni disponibili in due macro categorie:

- Tecnologie hardware native (che non richiedono lo z/VM);
- Tecnologie basate su z/VM.

Le prime sono basate su interfacce fisiche quali le schede OSA Express2 (in fibra o rame) o gli Hipersocket. Questi ultimi sono peculiari degli ambienti mainframe e costituiscono canali in memoria gestiti dall'hypervisor nativo (PRSM) per implementare LAN ad alte prestazioni (bassa latenza e alta banda) tra istanze Linux contenute in un unico sistema. A partire dal System z10 gli hipersocket possono operare in modalità Layer2, permettendo il trasporto di protocolli non IP. E' importante notare che in un ambiente mainframe le porte di rete fisiche sono condivisibili nativamente (anche senza z/VM) tra le diverse immagini Linux, ciascuna delle quali può presentarsi in rete con una propria identità, pur condividendo lo stesso cavo.

Le opzioni di networking basate su z/VM sono riconducibili alle Guest Lan e alla variante VSWITCH - Virtual Switch. Le Guest Lan sono LAN virtuali gestite dall'hypervisor software. Rispetto agli hipersocket, che possono essere definiti fino ad un massimo di istanze per sistema, le Guest Lan possono essere definite in numero maggiore, in funzione delle risorse disponibili. Di contro sono caratterizzate da un maggior overhead per effetto dello strato software di implementazione. Un VSWITCH è fondamentalmente una Guest Lan a cui sono associate una o più interfacce di rete fisica (porte OSA Express). Ciò rende possibile integrare in maniera molto semplice segmenti di rete interni al mainframe (con guest costituiti da immagini zLinux virtuali) con un segmento di rete fisica esterna al mainframe (vedi figura 5).

A partire dalle versioni più recenti di z/VM i VSWITCH:

- possono operare a livello 2 dello stack OSI, permettendo a ciascun guest Linux di avere un proprio MAC – Medium Access Control address;
- possono aggregare le porte fisiche al fine di incrementare la banda di comunicazione tra il segmento di rete interno e quello esterno (link aggregation support).
- supportano il protocollo IEEE 802.1q per cui è possibile avere guest che appartengono a VLAN distinte, eventualmente estese a host fisici.

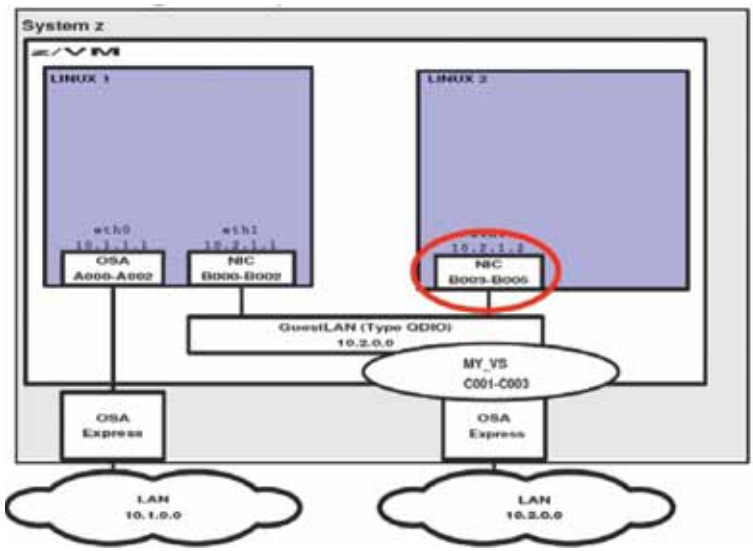


Figura 5 Connettività di Linux in z/VM

3.2.5 Linux nei Sistemi Virtualizzati con z/VM

La possibilità di ospitare diverse applicazioni, con i relativi livelli distinti dell'architettura, su un unico sistema scalabile contribuisce in maniera significativa a contenere molta della complessità legata al modello distribuito. La virtualizzazione si è imposta in questi ultimi anni come una delle strategie tecnologiche più promettenti per ottenere tali obiettivi di efficienza; in questo scenario si colloca l'utilizzo di zLinux e z/VM su ambiente mainframe.

In particolare, utilizzare sistemi Linux virtualizzati con z/VM comporta i seguenti vantaggi:

- completa condivisione di risorse tra molte immagini Linux sotto il controllo di un unico z/VM e conseguente "over commitment" delle risorse stesse;
- possibilità di consolidare molti "Serventi" Linux sullo stesso sistema;

- le immagini Linux possono trarre vantaggio dai Servizi offerti da z/VM e quindi utilizzare appieno le caratteristiche della z/Architecture;
- comunicazione efficiente attraverso le immagini, grazie alle connessioni interne, disponibile nella stessa macchina fisica (ad esempio hipersocket, Guest Lan);
- i dischi gestiti da z/VM possono essere suddivisi in Minidischi e condivisi tra le diverse immagini Linux;
- z/VM svolge una serie di funzioni avanzate tipiche della z/Architecture che non sarebbero disponibili sui Linux nativi (ad esempio incremento a caldo del numero di processori utilizzati dall'hypervisor);
- z/VM offre la possibilità di controllare e gestire le istanze da un unico punto di controllo e monitoraggio prestazionale.

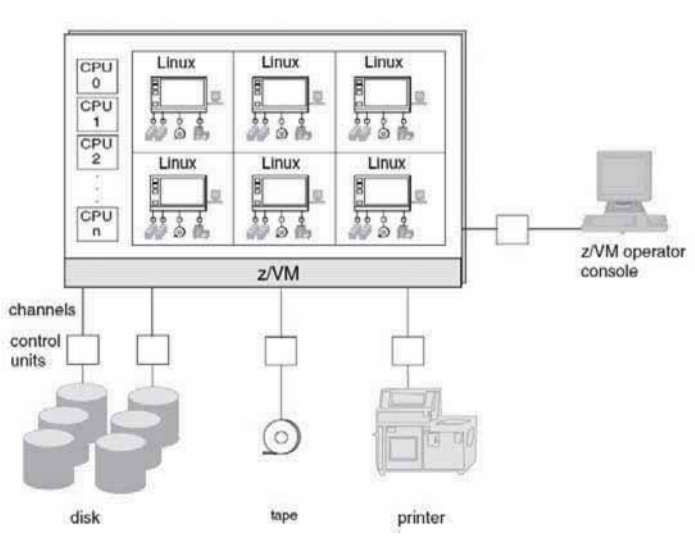


Figura 6 Guest Linux di un sistema z/VM

L'interazione tra z/VM e zLinux è stata progressivamente incrementata al fine di aumentare l'efficienza nella condivisione delle risorse hardware. Tra le tecnologie abilitanti sviluppate per implementare tale sinergia sono da segnalare:

- **Collaborative Memory Management Assist (CMMA)**, tramite cui z/VM e i guest zLinux scambiano informazioni al fine di ottimizzare l'utilizzo complessivo della memoria del sistema;
- **Virtual Machine Resource Manager (VMRM)**, che permette di controllare l'assegnazione delle risorse in contesa (ad esempio memoria fisica) tra gruppi di guest virtuali, in funzione della loro priorità.

3.2.6 Distribuzioni

Le distribuzioni disponibili su piattaforma mainframe sono sia di tipo commerciale (Novell SuSE e Red Hat), sia di tipo non commerciale (CentOS, Debian, ecc.). I distributori commerciali offrono servizi di supporto del sistema. Anche IBM offre suoi servizi di supporto[2].

3.2.7 Il processo di installazione

La maggior parte del processo di installazione di zLinux è simile a quello di Linux su altre piattaforme e dipende dalla specifica distribuzione utilizzata. Le differenze principali sono nel processo di avvio e nella configurazione dei dispositivi hardware specifici dei mainframe. Linux può essere installato direttamente su partizione LPAR o come ospite di z/VM (opzione più comune). E' possibile consultare, oltre alla documentazione della distribuzione scelta, le seguenti pubblicazioni IBM che forniscono esempi e "ricette" preconfezionate per l'installazione e l'utilizzo di zLinux su macchine virtuali:

- IBM z/VM and Linux on IBM System z: Virtualization Cookbook for Red Hat Enterprise Linux 5.2
  http://www.redbooks.ibm.com/abstracts/sg247492.html

- z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES 10 SP2
  http://www.redbooks.ibm.com/abstracts/sg247493.html?Open

In sintesi, nel caso di installazione su LPAR nativa, l'avvio del processo avviene mediante IPL dal lettore DVD della HMC. Nel caso di installazione su z/VM occorre copiare (ad esempio via FTP) i tre file per l'IPL (kernel, parameter file, ramdisk) su un guest z/VM del sistema (ad esempio CMS), spedire i file alla macchina virtuale target mediante virtual card writer[3] ed effettuare un avvio (IPL) dalla macchina virtuale target mediante virtual card reader. Gli step successivi per entrambe le modalità sono uguali e prevedono l'utilizzo di una sorgente di installazione in rete (server ftp, nfs, ecc.) che esponga i pacchetti di installazione (contenuto del DVD).

3.2.8 La clonazione di Sistemi Linux sotto z/VM

Per clonazione di un Sistema Linux si intende l'insieme di operazioni che consentono in poco tempo la creazione di una nuova immagine di Sistema Operativo Linux a partire da una "Copia Principale" esistente. La clonazione è possibile in quanto il Sistema Clone si differenzierà dalla copia principale o Golden Image solo per alcune configurazioni quali l'Hostname, gli indirizzi IP, le password, ecc. Tramite la clonazione è possibile rendere velocemente disponibili nuove immagini di Sistema in un ambiente mainframe, senza dover procedere a complessi passi di installazione.

---

[2] http://www-03.ibm.com/systems/z/os/linux/support_documentation.html

[3] I virtual card writer e reader sono dispositivi logici associabili a ciascun guest z/VM e sono utilizzati principalmente per la comunicazione tra guest.

Le operazioni da eseguire sono:

1. installare Linux per la prima volta, ad esempio usando un CD;
2. creare un Linux Template (Golden Image) da cui clonare le altre immagini; la Golden Image deve rappresentare un sistema minimo avviabile;
3. copiare la Golden Image per creare la nuova immagine (ad esempio effettuando copie fisiche dei DASD);
4. modificare gli opportuni file di configurazione del clone (hostname, indirizzo IP, ecc.);
5. modificare opportunamente le definizioni di z/VM per poter ospitare il clone.

z/VM facilita le operazioni di creazione di cloni, in quanto mette a disposizione una serie di programmi di utilità per copiare dati all'interno e tra macchine virtuali, per creare automaticamente nuove macchine virtuali e per avviarle e disattivarle automaticamente. Su un Sistema z/VM l'operazione di clonazione, compreso il riavvio, se opportunamente programmata, può richiedere pochi secondi. E' anche possibile clonare sistemi zLinux installati su LPAR nativa (vedere link pubblicazione).
http://www.redbooks.ibm.com/abstracts/redp3871.html

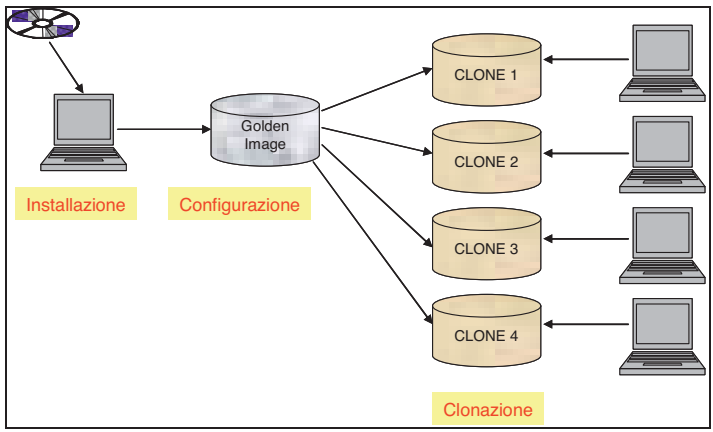


Figura 7 Clonazione di sistemi zLinux

3.3 Il Sistema Operativo z/VSE

In un sistema mainframe, oltre ai sistemi operativi già illustrati (z/VM, z/OS e zLinux), può essere utilizzato un altro sistema operativo, che, pur affondando le proprie radici in tempi lontani, è ancora attivo e ampiamente in uso presso moltissimi clienti IBM con elaboratori mainframe di capacità intermedia, sia in Italia che nel resto del mondo: lo **z/VSE**.

Questo sistema operativo è affine allo z/OS, in quanto progettato per carichi di lavoro di natura prevalentemente gestionale, per gestire quindi via TP (Transaction Processor) una mole anche significativa di utenti interattivi e contemporaneamente per gestire elaborazioni batch anche pesanti. Con lo z/OS ha in comune la maggior parte dei sottosistemi, per quanto con funzioni e in versioni parzialmente diverse, come il CICS, il VSAM, il TCP/IP, i linguaggi e i database. Rispetto allo z/OS presenta tuttavia una maggiore semplicità di installazione e di gestione e anche un minor costo, tuttavia può comportare alcuni limiti per utilizzi intensivi da parte di aziende di grandi dimensioni, con numero di utenti molto elevato e database voluminosi, per i quali il sistema operativo più indicato rimane comunque lo z/OS. Anche lo z/VSE, come gli altri sistemi operativi z/VM, z/OS e zLinux, può essere installato e funzionare sia in una partizione nativa di un sistema mainframe che in una macchina virtuale opportunamente definita, configurata e resa disponibile in ambiente z/VM.

3.3.0 Storia dello z/VSE

Il sistema operativo z/VSE è in realtà la più recente denominazione di una famiglia di sistemi operativi nati nel 1965, poco dopo l'annuncio del primo sistema mainframe (Sistema IBM S/360, 7 aprile 1964) e successivamente implementati fino ad arrivare alle versioni attuali. La prima versione del VSE si chiamò semplicemente **DOS** (**Disk Operating System**) e all'inizio prevedeva il supporto per l'esecuzione di un solo programma per volta, naturalmente batch, in monopartizione, senza alcuna possibilità di interazione TP.

Anno	Nome	Versione
1965	DOS	Disk Operating System
1973	DOS/VS	Disk Operating System/Virtual Storage
1979	DOS/VSE	Disk Operating System/Virtual Storage Extended
1984	VSE/SP	Virtual Storage Extended/System Product
1990	VSE/ESA	Virtual Storage Extended/Enterprise System Architecture
2000	z/VSE	Virtual Storage Extended for zSeries

Figura 1 Riepilogo delle principali denominazioni e versioni del VSE dal 1965 ad oggi

Nel 1973, con la versione **DOS/VS** (**Disk Operating System/Virtual Storage**) fu annunciato il supporto per il VSE della memoria virtuale che, grazie alla nuova tecnica della paginazione su disco, permise finalmente di appoggiarsi a un dataset di paginazione più ampio rispetto alla memoria reale effettivamente disponibile e quindi di ottenere un reale ed efficace multitasking. Nel frattempo venivano progressivamente introdotti prodotti integrativi per la gestione degli spool (Power), del TP (CICS), degli archivi (VSAM) e dei database (DL/1), oltre ai vari linguaggi di programmazione (Assembler e Cobol).

Pochi anni più tardi, nel 1979, con l'estensione del numero di partizioni e address spaces, assunse il nome di **DOS/VSE** (**Disk Operating System/Virtual Storage Extended**).

Nel 1984 il VSE, abbandonando definitivamente la storica denominazione di DOS, si configurò come un package con distribuzione e installazione semplificata e assunse il nome di **VSE/SP** (**Virtual Storage Extended/System Product**). Anche se l'indirizzamento rimaneva a 24 bit e quindi per ogni singola partizione restava il vincolo dei 16MB come dimensione massima (2^{24} = 16.777.216 bytes), era adesso possibile avere, grazie alla VAE (Virtual Addressability Extension), partizioni diverse in più address spaces, privati o shared, per un totale che poteva superare il fatidico ostacolo dei 16MB.

Nel 1990 venne annunciato il **VSE/ESA** (**Virtual Storage Extended/Enterprise System Architecture**) che permise il superamento di quasi tutte le limitazioni ancora presenti nell'architettura VSE e l'adeguamento alle caratteristiche dei più recenti modelli di sistemi mainframe.

Nel frattempo anche il VM, descritto in altra parte della presente pubblicazione, assumeva le caratteristiche dell'architettura ESA diventando VM/ESA e integrandosi sempre di più con il VSE, al punto tale che ormai per la maggior parte delle installazioni VSE era - ed è tuttora - contestualmente presente anche l'ambiente di virtualizzazione VM. La coesistenza di VM e VSE si dimostrava utile ed efficace per una migliore condivisione e una gestione ottimizzata delle risorse hardware disponibili (potenza elaborativa, memoria, spazio su disco), sia nel caso di un singolo sistema operativo z/VSE ospite, sia, anche e soprattutto, nel caso di più sistemi operativi z/VSE ospiti.

Il VSE/ESA è stata la versione che ha accompagnato i clienti VSE al cambio del secolo e al superamento delle relative problematiche (Y2K); nell'ottobre 2000, IBM annunciò l'architettura "z", che comprendeva elaboratori e relativi sistemi operativi interamente a 64 bit, rimuovendo definitivamente ogni residuo limite di indirizzamento.

Anche il VSE venne aggiornato con la versione **z/VSE** (**Virtual Storage Extended for zSeries**), che introdusse progressivamente il supporto per la nuova architettura.

In questi dieci anni lo z/VSE è ulteriormente cresciuto, fornendo sempre maggiori funzioni e migliori possibilità di collegamenti e connessioni, integrandosi sempre di più con gli altri sistemi operativi e mantenendo nel contempo la sicurezza, l'affidabilità e l'integrità tipiche della piattaforma mainframe.

3.3.1 Architettura

Caratteristica saliente dello z/VSE è di essere un sistema operativo di veloce installazione, ideale per i sistemi gestionali con applicazione batch e online ma con enormi capacità di adattamento. L'occupazione attuale del sistema operativo z/VSE versione 4.2.1 è di due dischi di dimensione pari a 2 GB l'uno, denominati **DOSRES** e **SYSWK1** (sui quali viene comunque lasciato circa la metà di spazio libero disponibile per integrazioni dell'utente).

Se installato come ospite di uno z/VM, lo z/VSE può demandare totalmente la funzione di paginazione della memoria virtuale allo z/VM stesso, scaricandosi di una incombenza spesso onerosa in termini di performance e di spazio disco.

In un'area comune a tutte le partizioni, chiamata **SUPERVISORE,** risiede il nucleo dell' IPL e il "cuore" del sistema. Con delle operazioni chiamate **SVC** (Supervisor Call) le partizioni statiche (BG,F0..FB) e le dinamiche richiedono specifiche attività al supervisore, ad esempio la necessità di utilizzare risorse esterne (nastri, dischi etc).

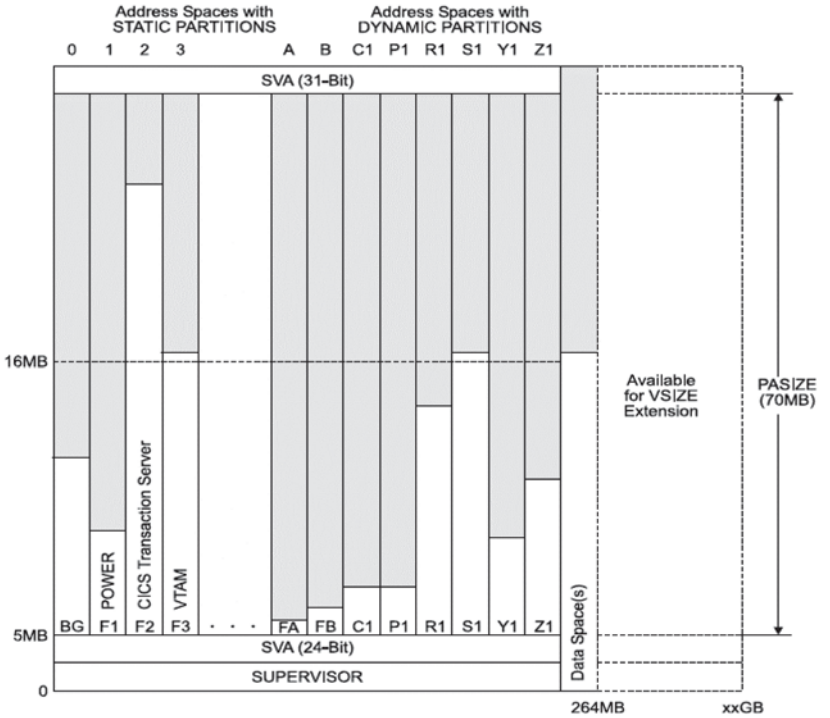


Figura 2 Esempio di struttura di memoria virtuale in z/VSE 4.2

Le parti di codice del sistema operativo o di programmi utente molto utilizzati possono essere caricate in memoria, differenziando però la memoria indirizzabile a 24 bit da quella indirizzabile a 32 (in realtà sono 31 vista la necessità di riservare un bit per esigenze di compatibilità) o a 64 bit.

Una delle caratteristiche principali degli ambienti mainframe è la possibilità di utilizzare su moderni sistemi anche comandi, utilities e programmi scritti per versioni precedenti di sistema operativo: non è difficile trovare degli ambienti VSE dove funzionano programmi scritti 30 anni fa.

In VSE esistono due aree di memoria virtuale, caricate al momento dell' IPL e utilizzate per evitare l'utilizzo di diverse copie dello stesso modulo in partizioni differenti, chiamate **SVA** (Shared Virtual Area): una che rimane sotto la "linea" dei 16 MB (programmi indirizzabili a 24 bit) e l'altra sopra i 16 MB (programmi indirizzabili a 31 bit).

Il sistema operativo z/VSE, oltre a essere installato su aree di disco formattate e indirizzate con modalità "proprietaria" (3380, 3390 etc.), può essere installato su aree di disco formattate con modalità SCSI (Small Computer System Interface), vedendo direttamente delle LUN (Logical Unit Number) di una SAN (Storage Area Network) esterna e indirizzate tramite uno specifico **WWPN** (World Wide Port Number).

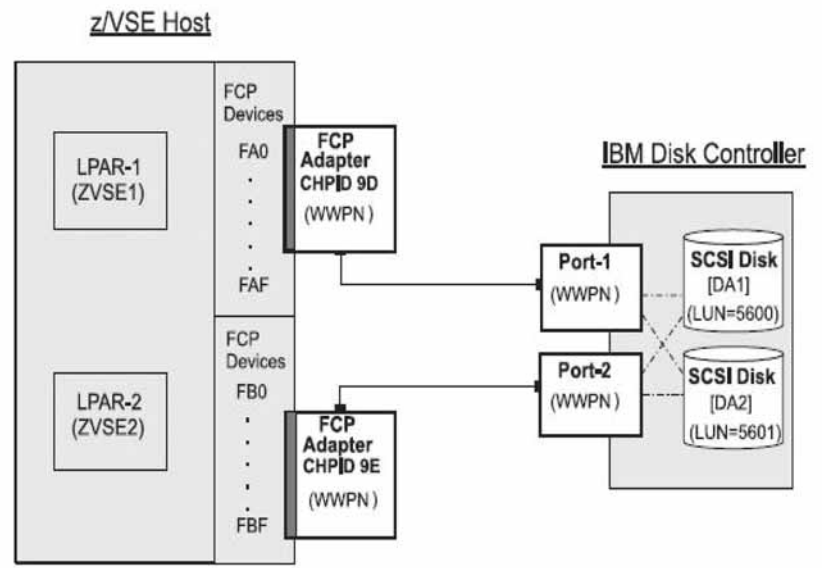


Figura 3 Esempio di collegamenti di un sistema operativo z/VSE con device SCSI su LUN indirizzate tramite WWPN

3.3.2 I prodotti

Come negli altri sistemi operativi, anche in VSE sono disponibili moduli e prodotti, base o opzionali, per realizzare specifiche funzioni e per gestire diversi sottosistemi.

Fra questi, i più importanti sono:

- **POWER** (**Priority Output Writers, Execution & Readers**) per la gestione degli spool e del flusso di dati in input e output;

- **CICS/TS** (**Customer Information Control System/Transaction Server**) sottosistema per la gestione dell'ambiente interattivo e il controllo delle risorse fra utenti TP, transazioni, programmi, files e database, etc. con relative funzioni di controllo, sicurezza, log e journaling;

- **ICCF** (**Interactive Computing & Control Facility**) editor 3270 interattivo ospitato nella partizione CICS e utilizzato negli ambienti VSE nativi (senza VM) dove non sono disponibili altri strumenti di editor più avanzati, quali CMS/XEDIT o TSO;

- **DITTO** (**Data Interface for Transfer, Testing & Operations**) utility per effettuare verifiche degli spazi disco, analizzare la VTOC e il contenuto dei dischi, realizzare copie da disco a nastro o fra unità a nastro anche di tipo diverso;

- **VSAM** (**Virtual Storage Access Method**) metodo di accesso per la gestione di files (sequenziali e a indici) non strutturati in forma di database;

- **SORT/MERGE** utility per ordinare e unificare archivi diversi, sulla base di specifici criteri forniti a programma o interattivamente;

- **DL/1** (**Data Language/1**) database manager per la gestione di banche dati strutturate secondo un disegno gerarchico;

- **DB/2** (**Data Base/2**) database manager per la gestione di banche dati in forma strutturata relazionale (con linguaggio **SQL**, *Structured Query Language*). In ambiente VSE il DB/2 è disponibile sia nella versione **server**, destinata a gestire e interfacciare database effettivamente presenti in ambiente VSE, che nella versione **client**, destinata a interfacciare dal VSE data base DB/2 effettivamente presenti in altri ambienti esterni rispetto al VSE (tipicamente Linux su motori IFL);

- **BSM** (**Basic Security Manager**) gestore della sicurezza e del controllo degli accessi al sistema, sia batch che interattivi;

- **VTAM** (**Virtual Telecommunications Access Method**) metodo di accesso alle reti di telecomunicazione di tipo SNA (*System Network Acrchitecture*) - che è una

architettura di rete locale e geografica precedente all'architettura TCP/IP, utilizzato per il controllo delle sessioni di terminali e stampanti, sia locali che remoti, per la gestione della rete, delle linee e delle interconnessioni con altri sistemi remoti; è ormai quasi interamente sostituito dalle funzioni Open del TCP/IP;

- **TCP/IP** (**Transmission Control Protocol/Internet Protocol**) protocollo di rete utilizzato per collegare il VSE con altri sistemi remoti - tramite scheda OSA-*Open System Adapter* per reti ethernet - o con sistemi interni al mainframe, quali, ad esempio z/OS, VSE, VM e Linux - tramite specifici canali di collegamento in memoria chiamati *Hipersockets* ;

- **COBOL** (**COmmon Business Oriented Language**) linguaggio di programmazione orientato ad applicazioni tradizionali di carattere prevalentemente gestionale, alla data è il linguaggio più utilizzato al mondo (è stato calcolato che ogni anno vengono scritte 5 milioni di righe di nuovo codice COBOL);

- **ASSEMBLER** linguaggio di programmazione molto vicino al codice macchina, ormai usato solo per la realizzazione di tools e utilities specifiche per cui sono richieste elevate performance;

- **C** linguaggio di programmazione nato per ambienti di tipo distribuito che è stato reso disponibile anche sulla piattaforma mainframe in ambiente VSE**;**

- **PSF** (**Print Services Facility**) prodotto che gestisce le funzioni avanzate di stampa - AFP (Advanced Functions Printing) per creare layout personalizzati integrando gli spool dei dati con immagini, loghi e fincature.

Oltre ai prodotti sopra elencati, forniti direttamente da IBM, sono disponibili sul mercato anche prodotti (Data Base Manager, Utilities, Linguaggi) - distribuiti da Società di Software e Terze Parti - specifici per l'ambiente VSE. Spesso questi prodotti integrativi hanno coperto delle aree funzionali che il sistema operativo principale e i relativi programmi opzionali avevano trascurato, diventando in effetti parte integrante del software di base; fra questi vanno considerati prodotti di schedulazione e automazione console, prodotti per la gestione dinamica degli spazi storage e per la gestione operativa di sottosistemi hardware robotizzati, librerie di nastri e virtual tape.

3.3.3 I connettori (z/VSE e-business connectors)

La possibilità di integrare il VSE con altri sistemi operativi o workstation ha contribuito al rilancio di questo sistema operativo, decisamente meno conosciuto dello z/OS o dello z/VM ma molto attivo nello sviluppo sia da parte di IBM che da parte di aziende di software.

Nel 1992 la società *Open Connect System, Inc.* (OCS), rilasciò il primo protocollo TCP/IP per i clienti VSE. Questa innovazione tecnologica fu realizzata utilizzando un server UNIX collegato al sistema VSE per eseguire il protocollo TCP/IP e dare la possibilità alle aziende di utilizzare FTP e TELNET. OCS a quel tempo serviva solo clienti del mercato MVS (il vecchio z/OS), così la società *Systems IntelliWare, Inc.* Di Dallas divenne partner della OCS per fornire le funzioni del protocollo TCP/IP ai clienti del mercato VSE come fornitore esclusivo. *Systems IntelliWare, Inc.* fornì corsi di formazione sul protocollo TCP/IP per clienti VSE e installò presso i clienti la soluzione di OCS.

Qualche tempo dopo, IBM iniziò a proporre la soluzione di OCS come protocollo TCP/IP per i clienti VSE. Nel 1995 la società *Connectivity Systems, Inc.* debuttò sulla scena con una implementazione TCP/IP nativa nel VSE, il che significava che non c'era bisogno di un server UNIX per fornire questa importante connettività di rete.

Dopo non essere riuscito a convincere OCS a proporre una soluzione nativa VSE, *Systems IntelliWare, Inc.* decise di diventare l'unico ed esclusivo canale di vendita per la nuova soluzione nativa della *Connectivity Systems, Inc.* negli Stati Uniti e lo commercializzò e installò in centinaia di siti in tutto il Nord America.

Sempre nel 1995, *Systems IntelliWare, Inc.* creò la prima interfaccia basata su Windows per il sistema operativo VSE, noto come Win/VSE® e nel 1997 IBM stipulò un accordo per rendere il protocollo TCP/IP disponibile per i clienti VSE in tutto il mondo, mentre *Systems IntelliWare, Inc.* continuò a vendere e supportare il protocollo TCP/IP introducendo il Web/VSE® (ora denominato z/Web-Host®), il vero primo prodotto che portava le applicazioni 3270 in un browser Web, permettendo ai clienti VSE l'accesso alle applicazioni da qualsiasi luogo, senza alcuna modifica alle applicazioni esistenti.

Nel 2000 *Systems IntelliWare, Inc.* introdusse il primo protocollo SSL (Secure Sockets Layer) per garantire la comunicazione cifrata dal VSE con altre reti e applicazioni Web, utilizzando il loro apparato iCYA™ per la crittografia su mainframe.

Con la versione VSE/ESA 2.5 (29 settembre 2000) IBM rispose alla crescente richiesta di modernizzazione di questo sistema operativo creando dei programmi client/server basati su tecnologia Java, che davano la possibilità di accedere in tempo reale alle risorse VSE (console, VSAM, POWER, DL/1, LIBRARIAN, ICCF) da workstation o server di sistemi distribuiti; questo tipo di software è stato chiamato **z/VSE e-business connectors**, o semplicemente "connettori".

Nel 2003 i prodotti di *Systems IntelliWare, Inc.* sono stati riconvertiti e resi disponibili da *Illustro Systems International, LLC*. (Illustro) che ha continuato il suo percorso di sviluppo di innovazioni per i clienti z/VSE, a partire dal rilascio di z/XML-Host®, un prodotto che fornisce SOA e servizi XML per z/VSE e z/OS.

Nel 2005 Illustro ha introdotto il primo AJAX/Web Interface per clienti mainframe (sia z/VSE che z/OS), attraverso il suo prodotto z/Web-Host® e nel 2008 ha introdotto il primo monitor TCP/IP per z/VSE (z/IPMon®).

Attualmente il laboratorio IBM z/VSE di Böblingen (Germania) è molto attento alle richieste degli utilizzatori e continua ad aggiornare e a produrre software che, utilizzando il protocollo TCP/IP, risolve e facilita molte attività, dando spesso soluzioni fondamentali a problemi architetturali di collegamento.

Figura 4 Rappresentazione grafica dei connettori VSE

In z/VSE è disponibile la funzione **DB/2 Client** (non ancora disponibile in ambiente z/OS), che permette di accedere al **DB/2 UDB** (**Universal Data Base**) in ambiente Linux, sia su sistemi mainframe che su altri sistemi operativi distribuiti. In questo modo non è necessario aver alcun dato DB/2 memorizzato nel sistema VSE e di conseguenza, installando solo la funzione client, si possono ottenere performance migliori (nel caso di DB/2 UDB su zLinux) con costi nettamente inferiori.

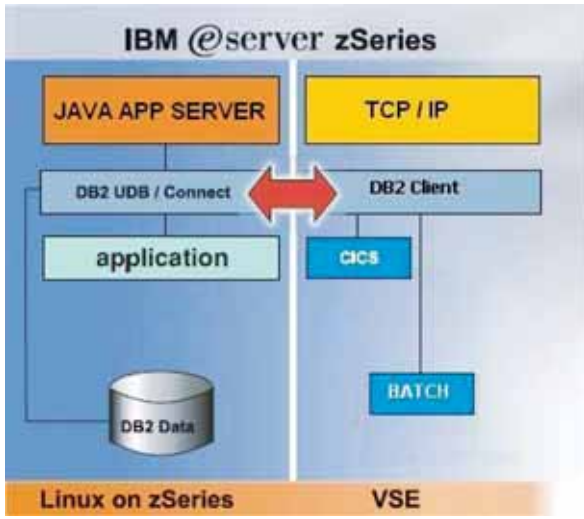


Figura 5 La funzione del DB/2 Client in z/VSE

La funzione dei connettori risponde sia alle esigenze dei sistemisti, per la manutenzione dell'ambiente, che alle esigenze degli utenti finali per migliorare la produttività e la qualità del lavoro. Una delle caratteristiche salienti dei connettori è quella di dare una vestibilità più moderna ai programmi di utilità del sistema, in modo da renderne più comprensibile l'uso senza dover ricorrere a degli editor proprietari del sistema mainframe (ICCF, XEDIT del CMS, etc.).

Ad oggi i connettori per sistema z/VSE resi disponibili da IBM sono :

- **VSE Script Server**, serie di comandi per poter accedere ai dati VSAM e alle risorse dello z/VSE da moderni programmi su workstation (ad esempio Microsoft Excel, Lotus);

- **VSE Navigator**, tool grafico per il controllo del sistema z/VSE, dalla console alle code del POWER (rdr, lst, pun), dai contenuti delle librerie a quelle dei file.
  Questo tool raggruppa le maggiori funzioni per la manutenzione delle risorse del VSE. Se ben configurato, utilizzando i prodotti di security opzionali o il Basic Security Manager VSE, permette di avere un unico prodotto di accesso al sistema per i programmatori, gli operatori e i sistemisti.

- **VSE Health Checker**, tool grafico per l'analisi delle performance del sistema z/VSE e dei suoi componenti base, utilizzato principalmente dai sistemisti;

- **VSE Map Tool**, tool grafico per la creazione di tracciati record di file VSAM da utilizzare con altri connettori di trasporto dati;

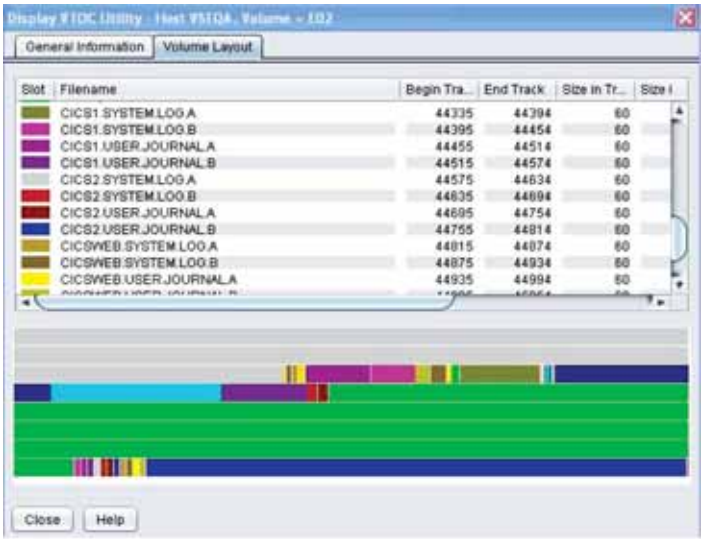


Figura 6 Esempio di utilizzo del connettore z/VSE Navigator per controllare la VTOC

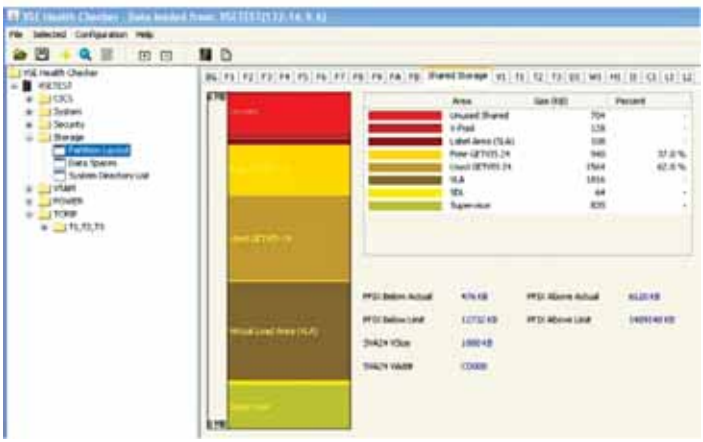


Figura 7 z/VSE Health Checker per l'analisi della memoria delle partizioni e della SVA

Peculiarità di questi quattro connettori è l'utilizzo di un programma JDBC client (**VSE Connector Client**) installato su piattaforma distribuita e di un programma server (**VSE Connector Server**) in ascolto in una partizione del sistema z/VSE.

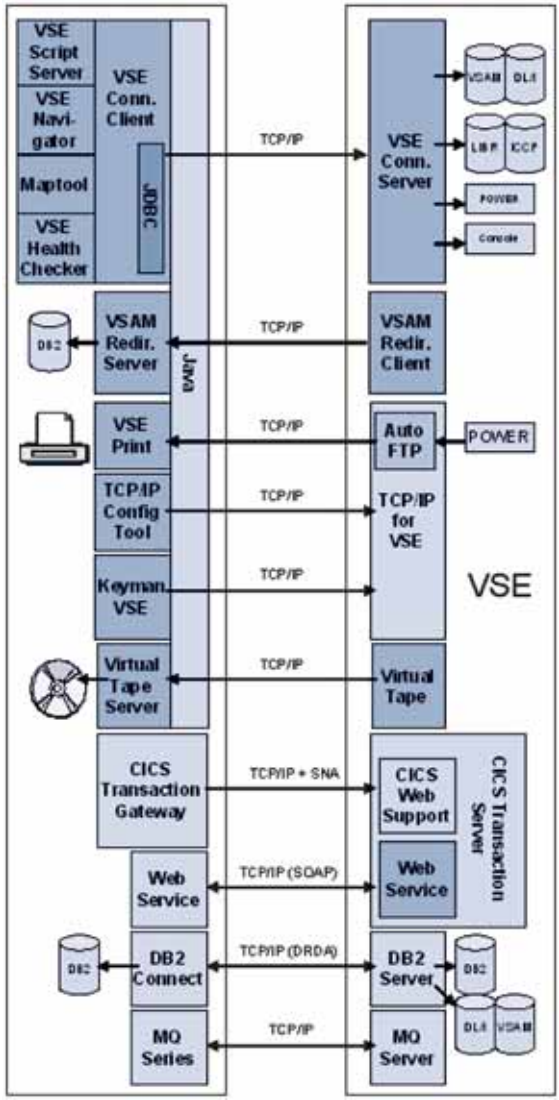


Figura 8 Schema generale delle funzioni TCP/IP in VSE

Fra le altre funzioni di connessione attualmente disponibili evidenziamo:

- **VSAM Redirector:** programma che permette di accedere a database, IBM o non IBM, su altre piattaforme (DB/2, Microsoft SQL Server, ORACLE) in emulazione di files VSAM. Questo connettore è stato creato per rispondere alle necessità degli utenti che vogliono integrare il sistema z/VSE con database su piattaforme diverse, senza modificare i programmi esistenti, quindi per poter accedere ai dati migrati su moderni database relazionali come se fossero dei file VSAM. Il VSAM Redirector necessita di una parte server su piattaforma distribuita e di un archivio nel sistema z/VSE con i dati di puntamento verso le tabelle relazionali di ogni file VSAM emulato. E' importante considerare che questo connettore si occupa di convertire il formato carattere ASCII (American Standard Code for Information Interchange) in **EBCDIC** (Extended Binary Coded Decimal Interchange Code) in andata e ritorno fra i due sistemi e che, oltre ai dati, converte i return code di accesso al database in return code propri di un sistema di accesso VSAM;

- **VSE print:** connettore tra il sistema POWER e le stampanti di rete, per poter stampare direttamente da z/VSE senza trasferire gli spool su workstation;

- **TCP/IP Config Tool:** tool grafico di edit del file di configurazione del TCP/IP;

- **Keyman/VSE:** tool per la creazione di certificati SSL o chiavi RSA – dai nomi degli inventori Ron Rivest, Adi Shamir e Len Adleman - compatibili con il sistema operativo VSE;

- **VSE System Class Library:** insieme di API (Application Programming Interface) in Java per l'utilizzo da ambiente remoto di specifici comandi in ambiente VSE;

- **Virtual Tape Server:** emulazione di unità nastro sotto forma di file su disco presenti su piattaforma distribuita. Molte installazioni di programmi o correzioni al sistema z/VSE possono essere fatte tramite la simulazione di utilizzo di unità nastro; prima dell'avvento di questo connettore l'unico input da esterno per il sistema VSE erano i nastri;

- **Web Service SOAP:** distribuito dalla versione VSE/ESA 2.7 (14 Marzo 2003), questa funzione ha reso possibile al VSE di richiedere servizi ad un WebServer o di fare da WebServer Provider basandosi sulla tecnologia XML con protocollo **SOAP** (Simple Object Access Protocol);

- **CICS2WS:** tool di creazione codice proxy per poter utilizzare il connettore Web Service SOAP .

Nel sistema operativo VSE sono disponibili due stack di TCP/IP distribuiti da IBM: CSI e BSI che sono prodotti da:

- **CSI** - Connectivity System International (distribuito di default da IBM)
- **BSI** – Barnard Software Inc.

IBM in ambiente VSE non ha proposto un proprio prodotto per il supporto del protocollo TCP/IP; questo inizialmente era valutato come un aspetto negativo dagli utilizzatori, abituati ad avere software di base interamente prodotto da IBM, ma con il tempo si è rilevato un punto di forza; oggi i due fornitori collaborano attivamente con IBM per il supporto di tutte le funzioni - supporto IPV6, NFS (Network File System), SSL, etc. - e propongono continuamente nuove funzioni per il VSE. IBM con gli z/VSE e-business connectors ha reso possibile il concetto di SOA (**Service Oriented Architecture**) anche in ambiente z/VSE.

La soluzione per la condivisione dei dati tra sistemi differenti, quindi il dato unico e non duplicato per incompatibilità di ambiente, è stata risolta con i client per DB/2, per MQ Series, con i VSAM Redirector o con la configurazione di un NFS, oltre a diversi programmi software disponibili sul mercato.

La possibilità di utilizzare un programma preesistente senza riscriverne il codice in un ambiente diverso o di richiamare lo stesso programma da diverse piattaforme, secondo le specifiche dell'architettura SOA, è stata resa possibile dall'utilizzo in VSE di un architettura Web Service SOAP utilizzando il **CICS/TS.**

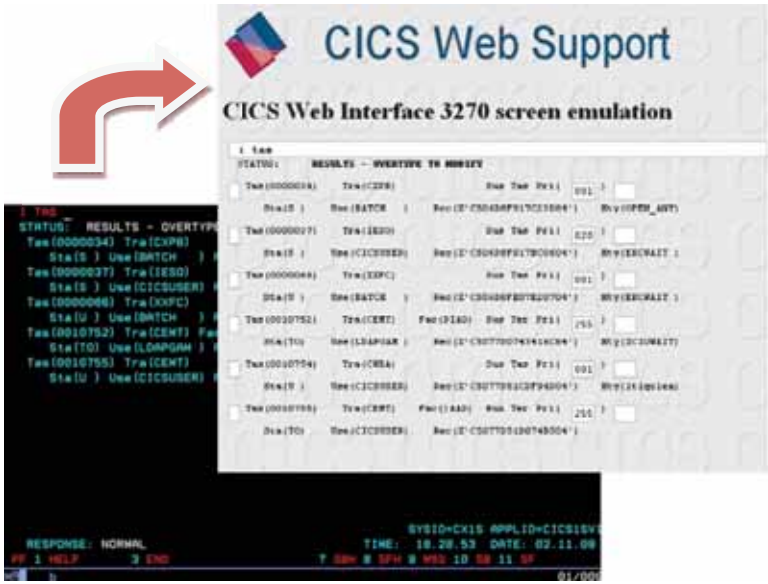


Figura 9 CICS Web Interface applicata ad un comando di sistema

3.3.4 La sicurezza in ambiente z/VSE

Lo z/VSE supporta un sistema di sicurezza di base, il BSM (Basic Security Manager) tramite il quale è possibile controllare gli accessi alle singole risorse del sistema (librerie, archivi, transazioni CICS, job batch).

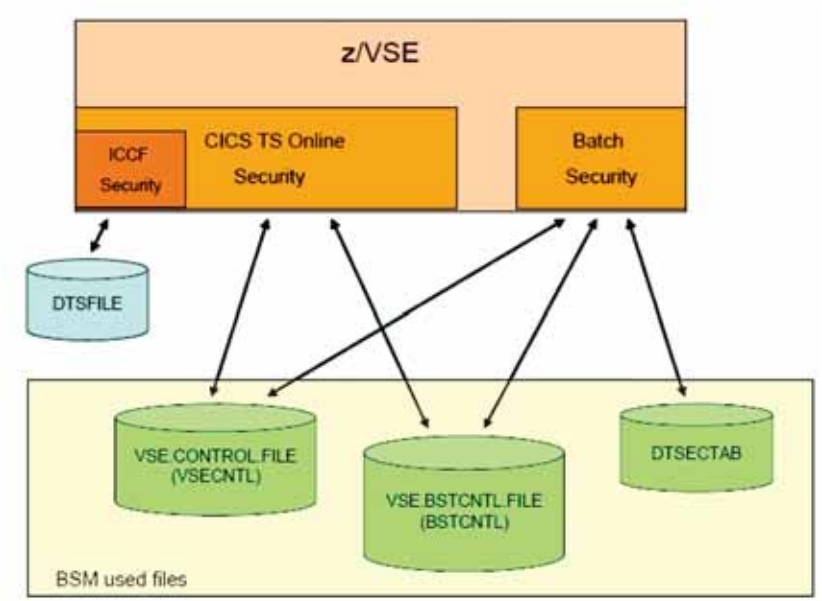


Figura 10 Rappresentazione grafica del BSM dello z/VSE

Oltre al **BSM** è possibile utilizzare prodotti di terze parti, come il "Top-Secret" di Computer Associates o "Alert" di CSI, che si sostituiscono totalmente alla sicurezza di base con dei tool che rendono più semplice l'attuazione di politiche di controllo, semplificandone la gestione.

Riguardo l'area dell'auditing, il sistema offre uno strumento, il DMS (Data Management System) che permette di creare un report con tutti gli accessi e i tentativi di accesso che sono stati effettuati sul sistema VSE.

Inoltre, tramite i connettori è possibile inviare automaticamente il report degli accessi a un unico sistema di registrazione degli accessi aziendali nel quale confluiscono i log di tutti i sistemi, in questo modo è possibile avere in una singola console la situazione in tempo reale degli accessi al sistema.

Nonostante il CICS possa gestire autonomamente le utenze registrate e gli accessi regolati tramite i diversi sistemi di sicurezza, dalle ultime versioni di z/VSE è possibile delegare l'autenticazione degli accessi, tramite il protocollo LDAP (Lightweight Directory Access Protocol), ad un sistema unico e remoto tipo IBM RACF (Resource Access Control Facility) disponibile in ambiente VM o Microsoft Active Directory.

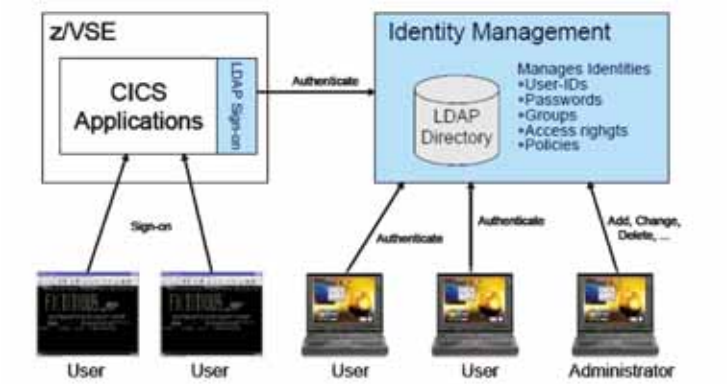


Figura 11 Condivisione della Directory LDAP dalle ws e dai CICS-TS del VSE

In questo modo si riesce a rispondere con una unica regola aziendale alle esigenze più moderne di tipologia di password (lunghezza minima superiore a 8 caratteri, case sensitive, etc.) senza dover utilizzare degli alias per l'accesso di lunghezza massima di 8 caratteri (vincolo non sempre di facile superamento in un ambiente mainframe).

Figura 12 Attuale accesso al CICS-TS dello z/VSE in modalità LDAP

3.3.5 Scenari di comunicazione e utilizzo

La caratteristica dello z/VSE di essere un sistema operativo molto leggero rispetto allo z/OS, sottolinea la possibilità di utilizzarlo come macchina virtuale ospite dello z/VM, permettendo di organizzare l'ambiente operativo secondo i migliori standard di sicurezza e rispettando le canoniche regole di duplicazione degli ambienti per attività di test.

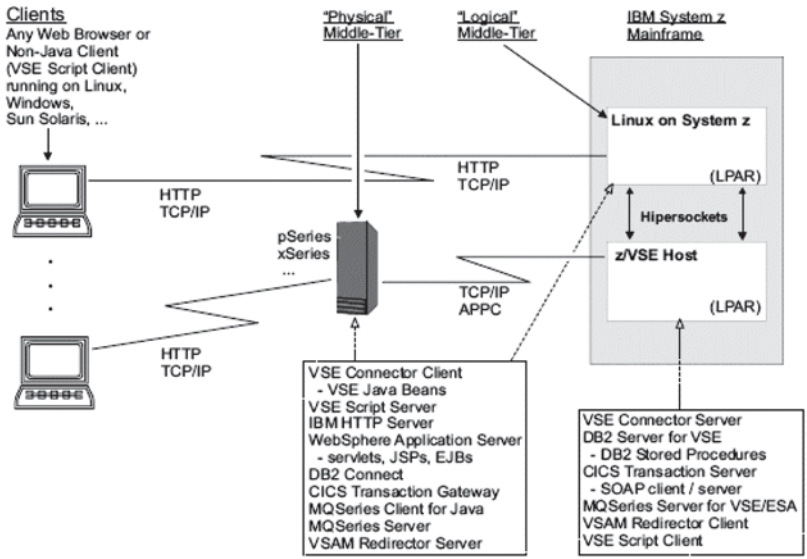


Figura 13 Esempio di interconnessioni di sistemi distribuiti con sistema mainframe in ambiente z/VSE e Linux.

Consideriamo, ad esempio, di avere un server IBM System z10 con due motori (CP e IFL) e un unico z/VM che gestisce diversi sistemi z/VSE e zLinux in un' unica LPAR.

Il collegamento tra i diversi sistemi operativi può essere gestito tramite la connessione di rete virtuale *hipersocket*, senza che le informazioni transitino fisicamente sulle schede di rete dello z10; mentre il collegamento tra i sistemi operativi mainframe e i sistemi "esterni" può essere gestito configurando in ciascun sistema operativo ospite dello z/VM una rete ethernet che fisicamente utilizza le schede OSA (Open System Adapter).

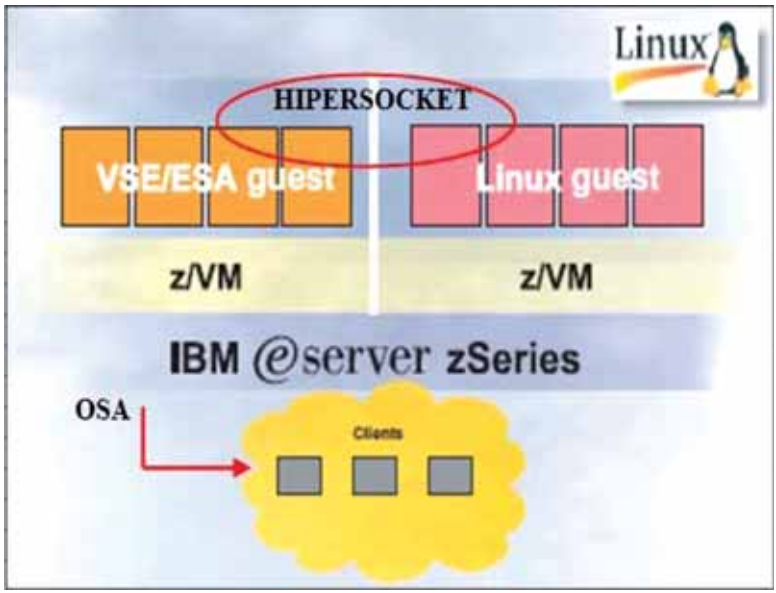


Figura 14 Rappresentazione grafica delle connessioni OSA e HiperSocket

In questo modo è utile avere un sistema z/VSE di test, uno di pre-produzione e uno di produzione.

La clonazione di un ambiente z/VSE tramite le funzionalità di FlashCopy, se abilitate nei moderni sistemi storage (ad esempio IBM DS8000), permette con semplici comandi z/VM di copiare istantaneamente i 2 dischi di base del sistema operativo VSE (DOSRES e SYSWK1) e quindi di creare velocemente altri ambienti VSE dedicati a prove, test e installazioni di nuove versioni.

Anche il VSE supporta, in modo nativo, i comandi di FlashCopy utilizzati per copiare archivi o dischi.

Tramite regole di firewall si possono attivare in VSE precise politiche di accesso in modo che nei tre sistemi citati (test, pre-produzione e produzione) i collegamenti siano mirati in base alle precise esigenze delle utenze e secondo regole di sicurezza predefinite. Così facendo anche i collegamenti tramite gli e-business connectors fra i VSE e i Linux interni al mainframe possono essere divisi secondo politiche di test e produzione.

3.3.6 Previsioni di sviluppo per le prossime versioni

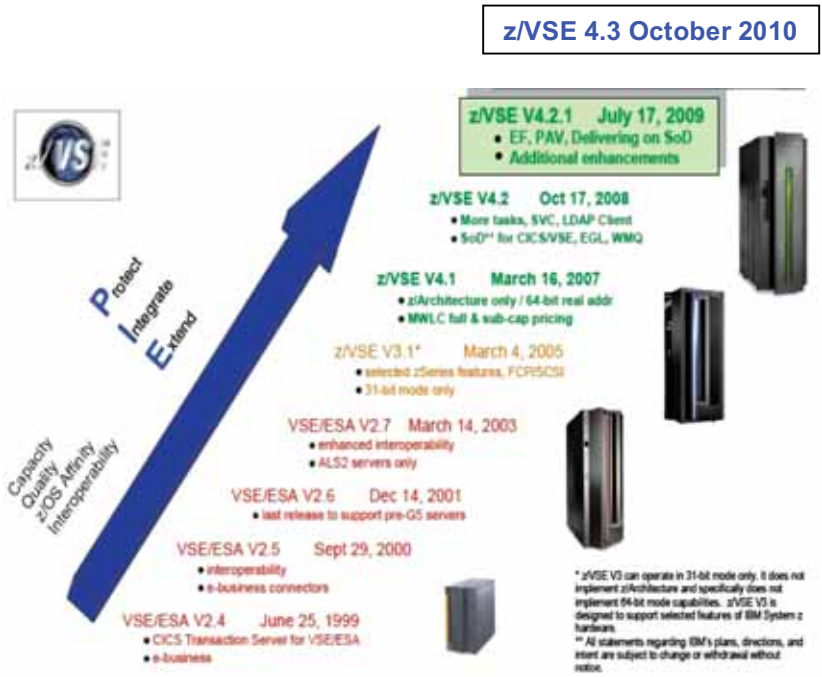


Figura 15 L'evoluzione delle più recenti versioni dello z/VSE con il concetto base di PIE (Protezione degli investimenti, Integrazione con altri dispositivi o sistemi, Estensione delle funzionalità)

In questa figura notiamo che le modifiche al sistema operativo z/VSE sono apportate di media una volta all'anno. Un tale sviluppo è confrontabile solo con altri sistemi operativi mainframe o con i moderni sistemi Linux, dove a proporre il codice è la comunità di tutti gli utilizzatori. Per poter modificare in tal modo un sistema operativo è necessario che gli sviluppatori siano sempre molto attenti all'evoluzione del mercato e alle esigenze delle utenze. Il team IBM dello z/VSE di Böblingen è sempre molto attento e aperto a questo genere di richieste di innovazione tecnologica.

In un ambiente mainframe per garantire la continuità del servizio è indispensabile un contatto diretto con il fornitore del software di base e del supporto alla manutenzione del prodotto, per il quale esistono diverse forme contrattuali che l'utente adatta alle proprie esigenze. Per esempio, nell'aprile 2011 l' IBM terminerà di supportare la versione z/VSE 4.1, quindi se qualche utente avrà un problema con tale versione del sistema operativo verrà comunque accettata la richiesta di aiuto ma se il problema non è documentato non verrà sviluppato nuovo codice su quella versione di VSE;

mentre se l'errore fosse su un sistema z/VSE 4.2 IBM si impegnerebbe a sviluppare il nuovo codice e la patch di riferimento.

Possiamo quindi affermare che in media IBM continua a sviluppare modifiche di una versione per quattro anni, anche se nel frattempo ne sono state rese disponibili altre.

In particolare, le principali innovazioni del sistema z/VSE 4.3.0, che sarà disponibile da ottobre 2010, saranno dedicate alla gestione sempre più efficiente dei nuovi hardware IBM, sia come sistema operativo nativo che in ambito z/VM come macchina virtuale. Inoltre ci sarà la possibilità di utilizzo di nuove funzioni direttamente da z/VSE su sistemi storage moderni, come i sistemi dischi DS6000 o DS8000, e di integrazione con altri software per il monitoring dell'ambiente (ad esempio **z/VPS**® di *Velocity Software*).

Figura 16 Il laboratorio IBM di sviluppo dello z/VSE a Böblingen (Germania)

4.0 Selezione della Piattaforma Informatica e Ottimizzazione della Infrastruttura

4.0.0 Introduzione

Scopo di questo capitolo è quello di fornire dei criteri oggettivi ed il più possibile quantitativi per la selezione della "Piattaforma Informatica" che deve ospitare (eseguire) una o più applicazioni informatiche. Tale processo ha senso in quanto le Applicazioni Informatiche sono oggi basate su linguaggi o strumenti di supporto alla esecuzione (Middleware) genericamente "portabili" su diverse piattaforme informatiche.

Partiremo dalle definizioni di Piattaforma Informatica ed Applicazione Informatica e introdurremo il concetto di Infrastruttura Informatica. Passeremo quindi alla individuazione di un metodo di Selezione basato su requisiti (Driver) di tipo non funzionale, ponendo attenzione al tema del Costo Totale della Piattaforma o della Infrastruttura. Per fare ciò introdurremo il concetto di equivalenza tra Sistemi di diversa Architettura e di Valore della Tecnologia Abilitante: questi due temi risulteranno fondamentali al fine del confronto.

L'infrastruttura risulterà il vero elemento del confronto da operare nel complesso, la sua ottimizzazione il suo fine ultimo.

Forniremo esempi della applicazione del metodo relativamente a tre casi reali: la determinazione della migliore piattaforma informatica per ospitare una nuova applicazione, la valutazione di un processo di cambiamento di piattaforma (re-hosting), il confronto tra piattaforme informatiche per un processo di Ottimizzazione.

Il tema della selezione della piattaforma per le applicazioni informatiche risulta di estremo interesse e si presenta ancora non completamente esplorato in tutti i suoi dettagli: ciò a causa della mancanza di criteri di confronto oggettivamente riconosciuti e condivisi e della pressione dei diversi Fornitori che tendono ad evidenziare soltanto alcuni aspetti (ai quali i loro prodotti risultano maggiormente aderenti), trascurandone altri, ugualmente importanti, ma non coperti da funzioni dei prodotti stessi. La visione complessiva che scaturisce da questa situazione finisce col confrontare elementi tra loro disomogenei, con il rischio di confondere il quadro complessivo del confronto.

Inoltre l'estrema peculiarità delle diverse soluzioni adottate dai Fornitori di Hardware e dai Progettisti di Sistemi Operativi (a volte sono soggetti diversi per lo stesso Sistema) rende in pratica impossibile la definizione di 'equivalenze' tra Sistemi. Considerare caratteristiche tipicamente HW (Clock del Processore, Velocità dei BUS, Banda di I/O), è facile ma fondamentalmente inutile quando si vuole stabilire una effettiva equivalenza fra i sistemi, cioè una formula che permetta di definire quanta potenza due sistemi differenti debbono impiegare per svolgere lo stesso carico di lavoro.

Le caratteristiche da confrontare potrebbero non necessariamente essere solo quelle prestazionali (più facilmente confrontabili) e le altre caratteristiche (ad esempio

Fruibilità, Ricoverabilità e Costo) potrebbero assumere un valore più rilevante nel confronto.

Perciò il compito di chi deve scegliere su quale Sistema eseguire una Applicazione non è affatto semplice: il primo problema da affrontare è quello di liberarsi da tutte le pressioni del Mercato, dei Fornitori, dell' "uso comune" e del "comune sentire", sempre più presenti nell'ambito della Tecnologia della Informazione, e peraltro in grado di inquinare qualunque confronto possibile. Successivamente dovrà dotarsi di un metodo, definire dei "driver" per il confronto, stabilire delle "metriche" e infine assegnare dei valori quantitativi in grado di determinare una "matrice di confronto".

Infine una precisazione: abbiamo assistito spesso al posizionamento dei Sistemi rispetto ai temi applicativi, ad esempio, nei primi anni duemila i Sistemi IBM di architettura Power RISC erano stati denominati Web Server, ad indicare una "vocazione" di tali Sistemi per applicazioni di tipo Web; qui intendiamo superare completamente il concetto di posizionamento per andare verso quello più complesso ed articolato della Selezione. Ciò nella convinzione, confermata dalle esperienze degli ultimi cinque anni, che le diverse Piattafome Informatiche possono essere indifferentemente utilizzate per qualunque tipo di Applicazione, cioè non sono connesse in alcun modo ai temi applicativi, e che la selezione debba essere fatta in base a criteri specifici di tipo non funzionale, come ad esempio il Costo, la Facilità di Esercizio, la Resilienza e le capacità di Continuità Operativa.

Il concetto di posizionamento, che poteva rivelarsi utile alla fine dello scorso millennio, quando alcuni strumenti erano disponibili solo su certi Sistemi Operativi, è oggi, dopo circa dieci anni, decisamente superato, anche grazie alla accresciuta portabilità reale delle Applicazioni sulle diverse Piattaforme Informatiche.

Assumeremo che ogni applicazione informatica possa essere attivata almeno su due piattaforme informatiche; nei casi in cui ciò non possa essere fatto diremo che ci troviamo in una situazione di 'legame esclusivo' tra la piattaforma e l'applicazione; in questi casi, che risultano sempre meno frequenti, la Selezione è un processo infattibile, a meno che non si pensi di riscrivere o portare l'applicazione sulla nuova Piattaforma, ma ciò richiede in generale un grosso sforzo che dovrà essere valutato. Infine parleremo di Infrastruttura Informatica e della sua ottimizzazione.

4.0.1 Definizioni

- Applicazione Informatica [**Applicazione**]: un programma o un insieme di programmi e procedure in grado di risolvere un problema di Business mediante l'uso di risorse computazionali (elaboratori elettronici), di memorizzazione ed apparati accessori.

- **Middleware:** un programma o un insieme di programmi di aiuto alla esecuzione della Applicazione Informatica, in grado cioè di fornire un ambiente di esecuzione controllato ed omogeneo con alcune funzioni di base, quali la gestione della interfaccia utente sul videoterminale, del controllo del flusso delle transazioni, della gestione degli errori e delle comunicazioni.

- Gestore dei Dati o Database Management System [**DBMS**] l'insieme dei programmi preposti alla gestione, al controllo e alla consultazione dei dati. Normalmente i DBMS sono di tipo relazionale. Compiti collaterali del DBMS sono quelli di garantire i modi di accesso al dato (sicurezza, autenticazione) e la loro congruità ed integrità (accessi condivisi, interruzioni e ripartenze).

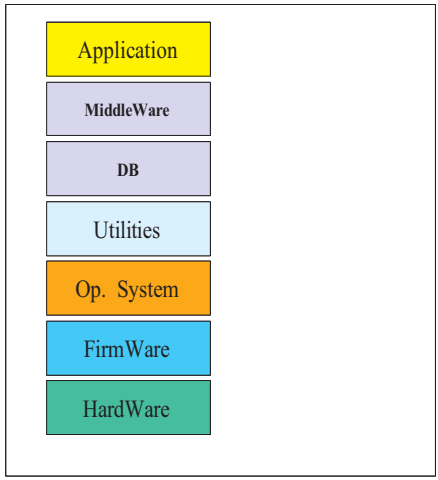


Figura 1 Lo Stack per l'Applicazione Informatica

- Programmi di Utilità [**Utilities**] un insieme di programmi, spesso facenti parte del Sistema Operativo o strettamente connessi ad esso, che svolgono funzioni di servizio, come ad esempio il controllo dell'accesso al Sistema, la sicurezza, le copie dei dati ed il monitoraggio del Sistema. Un particolare tipo di utility è costituita dai compilatori, cioè dai traduttori dei linguaggi evoluti in linguaggio macchina.

- Sistema Operativo **[Op System**] l'insieme dei programmi di base necessari al funzionamento e alla gestione efficiente del CEC. Fanno parte logicamente del Sistema Operativo anche i programmi che creano l'ambiente di esecuzione degli altri programmi, gestiscono gli errori e controllano la multiprogrammazione, cioè l'esecuzione simultanea di diversi programmi.

- **Firmware** una serie di programmi che servono per la gestione ed il controllo dell'Hardware. Un tipo di Firmware è quello che gestisce le risorse di Input/Output [I/O] ed un altro tipo è quello che gestisce i Sistemi Virtuali, ovvero quelli derivati da un unico Sistema Fisico scomposto in parti.

- **Hardware** il complesso elaborativo e di calcolo costituito da una o più unità elaborative [microprocessori], una determinata quantità di Memoria ed un certo numero di dispositivi di I/O.

La "pila" delle componenti della infrastruttura necessaria al funzionamento della applicazione informatica è schematizzata nella Figura 1 e prende il nome di stack.

La figura 2 indica come aggregare le componenti dello stack in modo da introdurre tre nuovi importanti definizioni.

- **Architettura Hardware** Le componenti Hardware e Firmware che costituiscono un Sistema elaborativo sono progettate in modo da fornire a coloro che scrivono SW (di qualunque livello) un'interfaccia il più possibile definita e stabile. Ad esempio un insieme di microprocessori Intel Xeon Mp con il relativo Firmware costituisce l'Architettura denominata x86. Spesso la stessa architettura Hardware può essere realizzata con componenti diverse. Ad esempio il microprocessore Intel Xeon Mp ed il processore AMD realizzano entrambi l'architettura x86. Questo insieme di regole comprende varie sezioni: il Set di Istruzioni, la tecnica di codifica dei dati, come viene indirizzata la memoria, come si gestisce il traffico di I/O e altre cose ancora. Un esempio di Architettura HW è stata fornita in questo libro: la z/Architecture. Tra le architetture più note citiamo ancora la PA-RISC e la SUNSPARC.

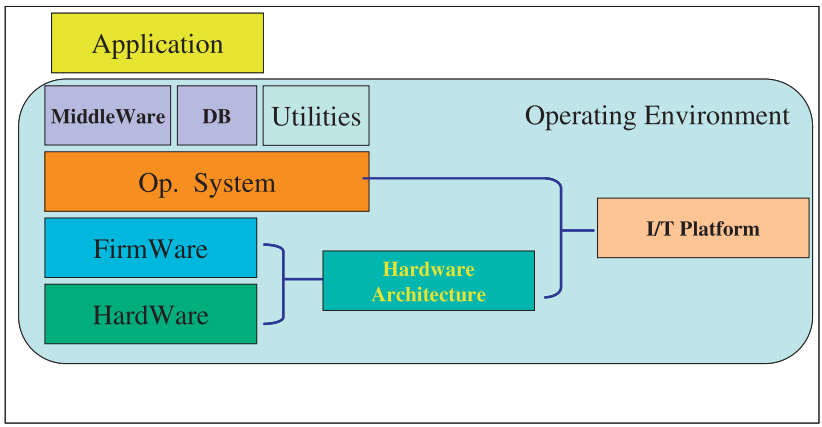


Figura 2 Piattaforma Informatica ed Ambiente Operativo

- Piattaforma Informatica [**Piattaforma**] la coppia ordinata di un Sistema Operativo ed una Architettura Hardware. Una tale definizione ci permette di identificare con precisione la coppia composta da Sistema Operativo ed Architettura Hardware poiché a volte lo stesso Sistema Operativo (o quanto

meno Sistemi Operativi con lo stesso nome e le stesse funzioni) può funzionare su Architetture Hardware diverse e la stessa Architettura Hardware può ospitare sistemi operativi diversi. Il concetto di **Piattaforma** risulta particolarmente utile tutte le volte in cui si vogliano identificare e confrontare ambienti diversi in grado di ospitare un'Applicazione Informatica.

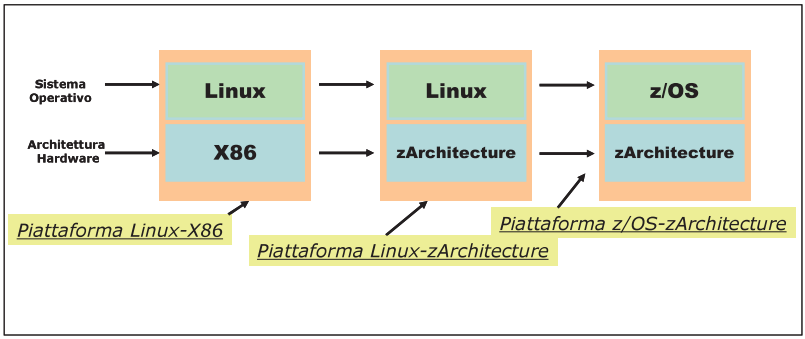


Figura 3 Esempi di Piattaforma Informatica

Con questa definizione un'Applicazione potrà essere eseguita (hosted) su diverse Piattaforme e quindi, a parità di Applicazione, avrà un senso il confronto e la selezione tra le Piattaforme possibili. La Figura 3 illustra il concetto di Piattaforma: l'Architettura Hardware denominata zArchitecture potrà ospitare Sistemi Operativi diversi, in questo caso z/OS e Linux. Nella denominazione della Piattaforma si usa mettere al primo posto il nome del Sistema Operativo seguito da quello dell'Architettura Hardware: ci sarà così la Piattaforma Linux-x86 ovvero Linux-zArchitecture, ecc.

Esempi di Sistemi Operativi:

- MS Windows
- Linux
- z/OS
- HP/UX
- Solaris
- IBM AIX
- ....

Esempi di Architetture Hardware:

- x86
- Apple
- zArchitecture
- PA-RISC
- Sun Sparc
- Power IBM
- ....

L'Architettura Hardware da sola o il Sistema Operativo da solo non bastano a definire la Piattaforma, poiché sulla medesima Architettura Hardware possono funzionare diversi Sistemi Operativi, o viceversa, lo stesso Sistema Operativo può essere ospitato su diverse Architetture Hardware.

Osserviamo che alcuni Sistemi Operativi possono essere attivati solo su una Architettura Hardware (ad esempio il Sistema Operativo z/OS può funzionare solo su macchine con zArchitecture), parimenti non tutte le coppie ordinate di un Sistema Operativo ed una Architettura Hardware danno luogo ad una Piattaforma esistente (ad esempio la Piattaforma Informatica MS Windows-zArchitecture NON ESISTE).

- Ambiente Operativo [**Operating Environment**] ogni possibile insieme coerente composto da una Piattaforma, Utilities, Middleware e DB relativo. Fissata la Piattaforma, l'Operating Environment può variare in base al Middleware, alle Utilities o al DB usato.

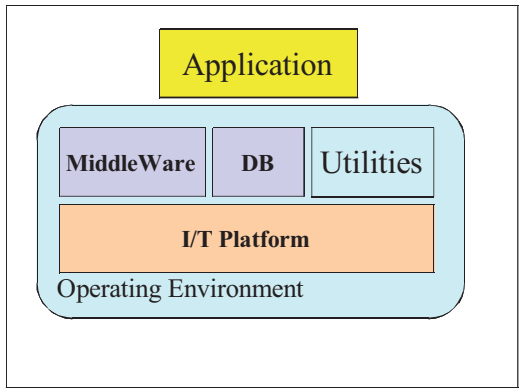


Figura 4 Operating Environment

Va detto che l'Applicazione di fatto si "appoggia" sull'Operating Environment - vedi figura 4 - che è il vero elemento differenziante: il processo di Selezione non dovrà fermarsi alla Piattaforma ma dovrà estendersi e tenere conto anche dell'Operating Environment.

Facciamo un esempio: Il Sistema Operativo denominato z/OS e caratteristico della piattaforma informatica [z/OS-zArchitecture] contiene un grande numero di Utilities integrate per gestire ad esempio la Sicurezza, il Movimento di dati, Le Copie di Riserva, il Monitoraggio del Sistema, ecc.; se accoppiato col DBMS relazionale IBM DB2 ed il Middleware IBM WebSphere realizza un Operating Environment completo per la esecuzione di applicazioni informatiche scritte col linguaggio Java: l'utente in pratica ha bisogno solo del Sistema Operativo z/OS, del Middleware WebSphere e del DB2; tale ambiente risulterà completo ed efficiente, e quindi di facile gestione.

Se la stessa Applicazione scritta in Java si volesse invece ospitare sulla Piattaforma Linux-zArchitecture (l'Hardware è lo stesso), sempre con WebSphere e DB2,

occorrerebbe munirsi di prodotti per gestire la Sicurezza, le Copie dei Dati, il monitoraggio, ecc., poiché il Sistema Operativo Linux non ha utilities sofisticate; l'Operating Environment sarebbe molto diverso, sia nelle operazioni che nella qualità del Servizio, da quello basato su z/OS.

Secondo l'esempio descritto la valutazione per scegliere la Piattaforma Informatica dovrà tenere conto anche delle diversità dell'Operating Environment a parità di Hardware, Middleware e DB.

Nell'ultimo decennio si è sempre più sviluppata la capacità dei Sistemi di Elaborazione di ospitare al loro interno diverse 'istanze' di Sistema Operativo, ciascuna indipendente dalle altre; tale capacità, sia pure con diverse caratteristiche, è presente su tutte le Piattaforme, e prende il nome di virtualizzazione. I Sistemi vengono detti allora **Sistemi Virtualizzati**.

In un Sistema Virtualizzato l'Hardware viene suddiviso in parti che possono essere fisiche o logiche, in ciascuna delle quali può essere attivata una istanza di Sistema Operativo. I Sistemi Virtualizzati possono avere partizioni logiche o fisiche; di norma su un Sistema Virtualizzato possono essere attivati tutti i Sistemi Operativi compatibili con l'architettura hardware del Sistema. Si ritiene che nel prossimo futuro potranno essere attivate sullo stesso Sistema fisico anche diverse Architetture Hardware, in tal modo più piattaforme informatiche (anche con diverse Architetture Hardware oltre che con diversi Sistemi Operativi) potranno essere attivate sul Sistema Fisico. Si parlerà in questo caso di "**Sistema Ibrido"**.

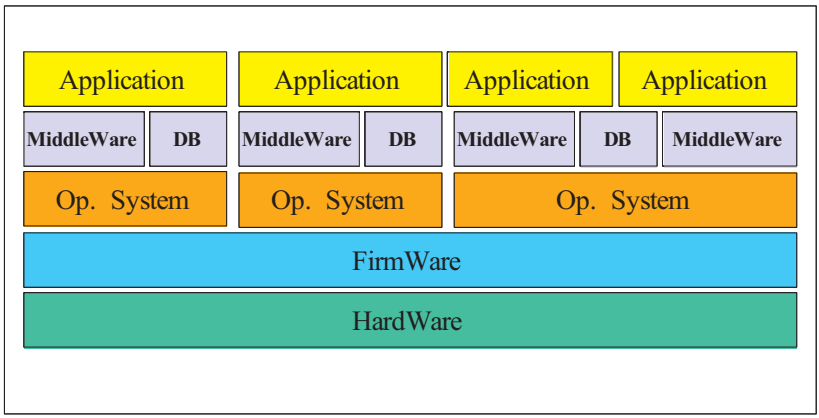


Figura 5 Sistema Virtualizzato

- **Infrastruttura Informatica:** l'insieme degli apparati Hardware (Serventi, Unità a Disco, Unità a nastro, Stampanti, ecc.), del Software, del Middleware, delle linee, delle strutture di Rete, delle altre interconnessioni e del Personale, utili per generare e gestire tutti gli Operating Environment necessari ad eseguire una Applicazione.

Più in generale l'***Infrastruttura Informatica è l'ambiente di esecuzione di una o più Applicazioni Informatiche.***

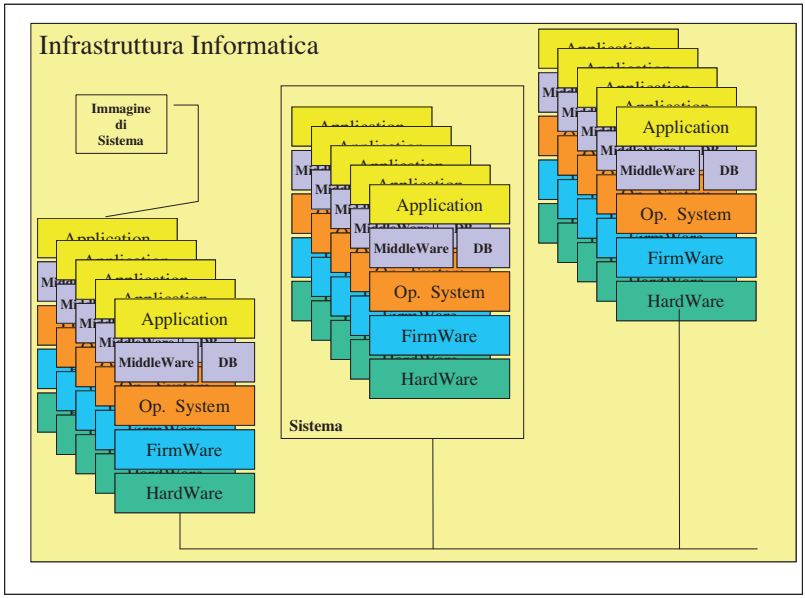


Figura 6 Infrastruttura Informatica dei Sistemi - Vista Logica

Fanno parte della Infrastruttura Informatica:

- i Serventi (Elaboratori Elettronici) con il loro Software di Base (Sistemi Operativi), Software e e Middleware;
- gli apparati di Memorizzazione su Nastro o su Disco Magnetico o altri supporti (CD, DVD, ecc.);
- gli apparati di comunicazione locali e remoti (Reti, Hub, Switch, Modems, ecc.);
- i terminali Video, i Personal Computers collegati in rete con i relativi Software di Emulazione;
- il Personale di Gestione dei Serventi, dello Storage e della Rete;
- altri Apparati hardware, come ad esempio le stampanti;
- tutti i programmi e le procedure che costituiscono l'Applicazione Informatica;
- i Dati gestiti dall'Applicazione.

Per come qui definita, nell'Infrastruttura Informatica potranno essere presenti una o più Piattaforme (Figura 6) ed inoltre una Infrastruttura Informatica potrà eseguire una o più Applicazioni. A causa del rapido sviluppo delle Applicazioni e della Rete, negli ultimi decenni l'Infrastruttura Informatica si è notevolmente complicata, tanto che oggi essa è diventata un elemento di grande complessità e criticità. In particolare, negli ultimi anni sono emerse alcune problemi specifici. Questi richiedono oggi degli

interventi mirati, atti a superare questi problemi e a rendere l'Infrastruttura meno critica e più semplice da gestire.

L'insieme degli interventi atti a rendere l'Infrastruttura Informatica più semplice, più stabile, più efficiente e più gestibile prende il nome di **Ottimizzazione della Infrastruttura Informatica** - ITRO (Information Technology Resources Optimization). La Figura 7 mostra una vista di insieme complessiva della Infrastruttura Informatica di una Azienda o Ente. Una Infrastruttura Informatica risulterà composta da diversi Sistemi Fisici, dei quali tutti o alcuni potranno essere Sistemi Virtualizzati residenti su Piattaforme differenti che realizzeranno uno o più Operating Environment diversi sui quali si appoggeranno una o più Applicazioni.

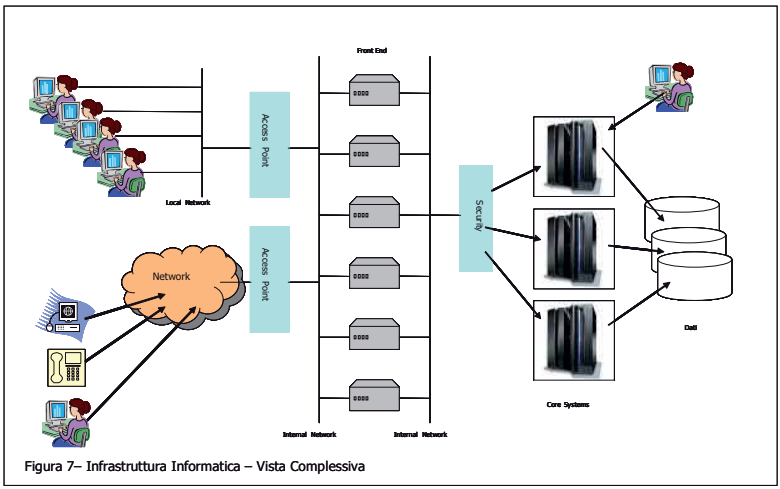


Figura 7 Infrastruttura Informatica - Vista Complessiva

4.0.2 Portabilità, Standard, Openness

Una **Applicazione** è un insieme di manufatti Software (Programmi, Procedure, Dati) in grado di svolgere un lavoro utile attraverso un Sistema di elaborazione; ad esempio è l'insieme dei programmi che gestiscono i Conti Correnti di una Banca, le Paghe di una Azienda, le prenotazioni di una Linea Aerea, la Segreteria Studenti della Università, i calcoli per la ricerca petrolifera, per i modelli di simulazione o per un problema matematico di grande complessità. Può essere eseguita su uno o più Operating Environment differenti; la stessa Applicazione potrà avere parti o componenti residenti su Piattaforme differenti contemporaneamente.
Si dovrebbe perciò dire che la scelta avviene fra Operating Environment più che fra Piattaforme; siccome è invalso il termine di Platform Selection continueremo a usare

quest'espressione, tenendo conto che la scelta vera avviene fra Operating Environment.
Scegliere implica l'esistenza di alternative: nel nostro caso l'alternativa esiste se l'Applicazione Informatica può essere attivata su almeno due Operating Environment diversi. Questa caratteristica è la portabilità.
Facciamo un esempio pratico. Abbiamo un'applicazione Transazionale Online basata sul DBMS SQL Server, disponibile solo sul sistema operativo Windows: in questo caso parleremo di applicazione legata al Sistema Operativo Windows. L'ambito di portabilità di questa applicazione sarà limitato esclusivamente all"architettura HW x86 e a quei fornitori HW che la implementano (INTEL, AMD).

Ai fini della Platform Selection si assume che un'Applicazione sia portabile se il codice sorgente che la compone può essere ricompilato in modo da poterla ospitare su un altro Operating environment. In tal modo la scelta sarà possibile. Si accetta che oltre alla fase di ricompilazione siano necessari piccoli aggiustamenti imposti dal nuovo Operating Environment.

Figura 8 Portabilità di una Applicazione Informatica

Per facilitare la portabilità è essenziale che i vari Operating environment aderiscano il più possibile ai cosidetti **Standard:** insiemi di regole che, definite da Organizzazioni internazionali o divenute tali "de facto", governano alcuni processi della elaborazione, come ad esempio le comunicazioni, l'organizzazione dei dati, la memorizzazione degli stessi, le tecniche di visualizzazione.

Gli Standard costituiscono un elemento utile alla portabilità delle Applicazioni sui diversi Operating Environment: gli standard *de iure* o *de facto*, come ad esempio TCP/IP, XML, Java, J2EE, si aggiungono ai Middleware (Apache, WebSphere, TomCat, ecc.) nel facilitare la possibilità di esecuzione (deploy) dell'Applicazione Informatica su diverse Piattaforme con lo stesso risultato funzionale e senza modifiche al codice.

Solo grazie alla portabilità ha senso definire un processo di selezione della Piattaforma Informatica - "Platform Selection", in quanto la stessa Applicazione Informatica, con modifiche piccole o nulle al codice, potrà essere eseguita sotto diversi Sistemi Operativi, o più frequentemente sotto lo stesso Sistema Operativo e diverse Architetture Hardware con diverse caratteristiche prestazionali o di costo. La platform selection implicitamente definisce anche il relativo Operating Environment e quindi le caratteristiche di gestione della Applicazione.

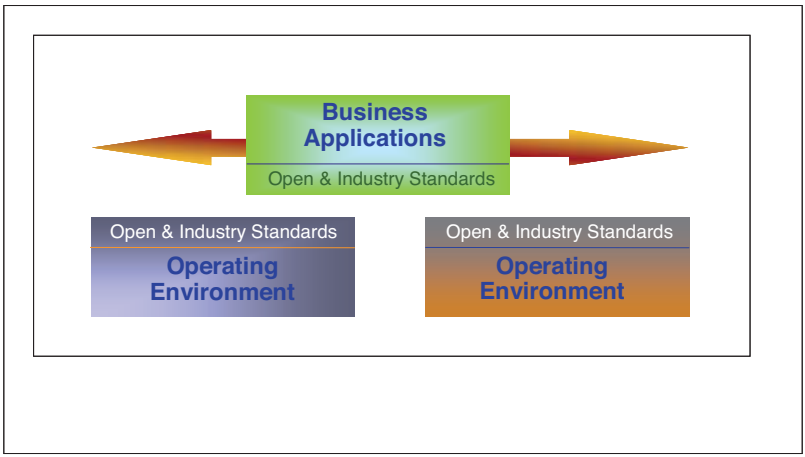


Figura 9 Il Ruolo degli Standard

Se il "porting" non fosse possibile non avrebbe senso parlare di Selezione[1].

Sembra importante a questo punto una riflessione sul tema degli Standard ed in generale della "Portabilità" e sui concetti che ne derivano di "Applicazione Aperta" [Open] o "Applicazione legata" [Legacy], in quanto i termini suddetti [Open e Lagacy] vengono spesso utilizzati in modo improprio o alterandone il significato.

Esistono **Standard di tipo formale** [de iure], ovvero definiti secondo un processo codificato da organismi internazionali a ciò preposti[2]; in teoria solo gli standard di questo tipo possono essere assunti al ruolo di formale "standard"; esempi di standard formali sono la codifica dei dati ASCII, la rappresentazione dei numeri IEEE Floating Point, il protocollo IEE 802.11 , lo standard ISO/OSI, etc. Il processo di formazione di uno Standard è lungo e complesso. Per loro definizione, essendo di dominio pubblico, gli standard non sono proprietà di nessuno, quindi chiunque può "produrre" secondo uno standard.

Esistono anche degli standard di fatto [de facto], ovvero non corrispondenti a definizioni formali e spesso corrispondenti ad oggetti proprietari ma talmente diffusi

[1] Il Tema della Platform Selection ha preso corpo da quando la Tecnologia della Informazione ha assunto in maniera diffusa l'uso di Standard e di Middleware multi- piattaforma. Fino alla metà degli anni '90 la Piattaforma Informatica era automaticamente determinata dalla Applicazione Informatica e non era pensabile alcuna possibilità di selezione.

[2] Gli organismi internazionali più importanti sono: la International Standard Organization [ISO], la **I**nstitute of **E**lectrical and **E**lectronic **E**ngineers [IEEE] e l'American National Standard Institute [ANSI].

da risultare "standard"; il TCP/IP ad esempio è uno standard di fatto nelle interconnessioni tra i calcolatori.

Gli standard di fatto più popolari sono ad esempio il linguaggio Java[3] che ha sostituito nei fatti il COBOL (che era uno standard formale della American National Standard Institute [ANSI], come il linguaggio C anch'esso standard formale ANSI), ovvero il Sistema Operativo Windows[4]. Essi sono stati elevati al rango di standard in quanto molto diffusi, ma non lo sono. L'affermazione che una applicazione **è standard** perché si basa sul Sistema Operativo Windows è priva di significato; in questi casi l'aggettivo standard è sinonimo di "molto diffuso" ma si riferisce comunque ad un prodotto "proprietario" e quindi soggetto a regole e controllo da parte di chi ne detiene la proprietà.

Osserviamo che uno standard "de iure" evolve sotto il controllo dell'organismo che ne pubblica le caratteristiche; uno standard "de facto" evolve sotto la spinta di varie esigenze (il mercato, gli sviluppatori, gli utenti). Uno standard de Facto può diventare de Jure se si costituisce un organismo "super partes" che ne controlli l'evoluzione. IETF (Internet Engineering Task Force) è un esempio di organismo che fa diventare alcune parti del TCP/IP standard de Jure attraverso il meccanismo degli RFC (Request for Comments – sono proposte di modifiche alle specifiche TCP/IP proposte a utenti e implementatori). Probabimente anche Java potrebbe subire la stessa evoluzione attraverso il meccanismo degli Java Specification Request (JSR) che svolgono la funzione degli RFC nel mondo TCP/IP.

E' necessario fare chiarezza sul concetto di Sistema Aperto o Open, che si contrappone a quello di Sistema Proprietario; ciò si riferisce al tema di Piattaforma Open o Proprietaria.

Poiché la piattaforma informatica contiene le componenti Hardware e Firmware che non sono mai standard (i microprocessori sono prodotti da chi li ha progettati o su licenza), essa sarà sempre in qualche modo proprietaria; ritenere ad esempio che l'architettura x86 sia Open o Standard non è corretto. Si può dire che ci sono più produttori che fabbricano microprocessori con architettura x86; ciò è vero come il fatto che negli anni ottanta vi erano più produttori che fabbricavano processori di architettura IBM S/390, e ancora oggi vi sono produttori che fabbricano processori con architettura SPARC. Ciò è dovuto al fatto che i progettisti nei tre casi citati (INTEL, IBM e SUN) hanno reso pubblica l'interfaccia esterna della architettura e il set di istruzioni, consentendo così ad altri di fabbricare processori internamente differenti, ma con la stessa interfaccia e lo stesso set di istruzioni. Il fatto che vi possano essere più produttori di una Architettura Hardware può dipendere dalla sua diffusione sul mercato; diffusione tuttavia è un elemento transitorio che non può essere considerato come Openness o Standard.

Diversa è la situazione dei Sistemi Operativi; se è vero che ne esistono molti di tipo proprietario (z/OS, AIX, Solaris, HP/UX), è altresì vero che ne esistono molti altri effettivamente Open (Linux, FreeBSD, OpenSolaris). Nel caso dei Sistemi Operativi il termine Open ha un significato preciso: definiti "Open Source", di questi sistemi operativi è pubblica la struttura interna ed il codice (istruzioni in linguaggio evoluto)

[3] Marchio Registrato della SUN Microelectronics .
[4] Marchio Registrato della Microsoft Corporation

con cui sono realizzati; in tal modo è possibile per chiunque conosca il linguaggio (in questo caso il linguaggio ANSI C, standard de iure) apportarvi modifiche o correzioni. Ciò non è solo una dichiarazione di intenti ma una realtà concreta che ha portato, per esempio, alla realizzazione di versioni del Sistema Operativo Linux per diverse architetture hardware (praticamente tutte), consentendo di definire piattaforme di tipo Linux per zArchitecture, Itanium, IBM Power, SPARC o parimenti di versioni di Open Solaris per zArchitecture ed x86. Ricordiamo che Linux era nato per x86 e Solaris per SPARC.

Quindi per i Sistemi Operativi il concetto di Open ha un senso preciso ed indica un Sistema Operativo **Open Source** del quale il codice sorgente è reso pubblico in tutte le sue parti.

Tale riflessione deve essere estesa a tutta la pila [stack] su cui si poggia l'applicazione (vedi Figura 10); ad ognuna delle componenti della pila, escluso l'Hardware ed il Firmware [cioè l'Architettura Hardware], dobbiamo estendere i concetti di Standard e di Openness, facendo particolare attenzione a non confondere il concetto di Standard de iure con gli Standard de facto ed avendo cura di estrarre da questi ultimi gli Open Standard. Con la stessa cura dovremmo scegliere le componenti Open Source e distinguerle da quelle percepite come standard, in quanto molto diffuse, ma nei fatti proprietarie.

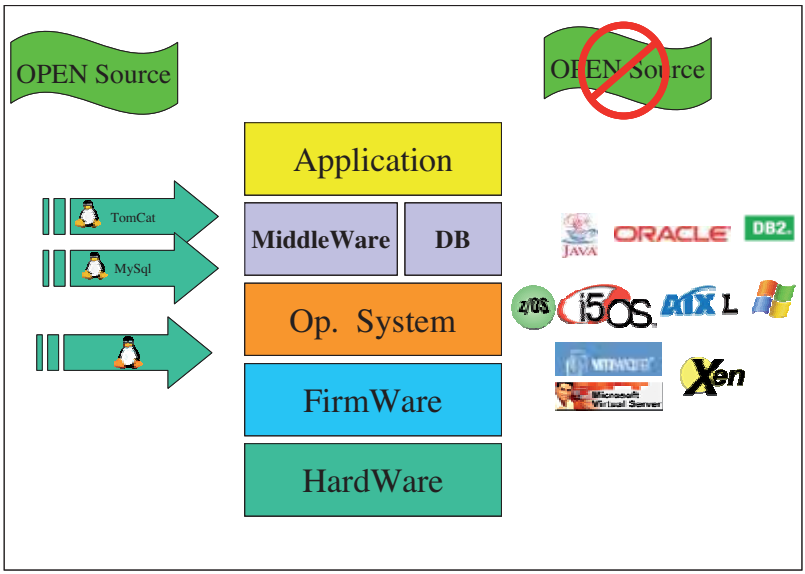


Figura 10 Open Source

Facciamo l'esempio di uno stack basato sul Sistema Operativo Linux, il prodotto Open Source MySQL come Gestore di dati, il prodotto Open Source JBOSS come gestore di applicazioni Java, il protocollo html per gestire la presentazione con il prodotto Open Source APACHE, il prodotto Open Source OPENLDAP per gestire gli accessi al Sistema ed il protocollo TCP/IP per le comunicazioni; questo stack può essere considerato Open anche se dovremo comunque escludere componenti Hardware e Firmware proprietarie sulle quali però esso risulterà portabile senza alcun intervento specifico.[5]

Non appena nel nostro esempio si cambia qualcosa, ad esempio se si sostituisce la componente MySQL con un RDBMS, come IBM DB2, che è disponibile per Linux su tutte le piattaforme informatiche, la soluzione non è piu totalmente Open. Ciò non vuol dire che la soluzione non possa funzionare, ma si vuole sottolineare il valore ed il significato del termine Sistema Aperto in contrapposizione col termine Sistema Proprietario.

I termini Standard e Open vengono usati spesso in senso positivo in contrapposizione col termine proprietario che viene invece usato in senso negativo. Questa contrapposizione è priva di senso: non è vero né è provato dai fatti che Open sia la soluzione giusta e proprietario quella sbagliata; tale retaggio è solo di tipo psicologico e nasce dalla contrapposizione che abbiamo vissuto negli anni '80 tra alcuni fornitori ritenuti "monopolisti", che venivano definiti "not open" dai fornitori emergenti; questi proponevano soluzioni nuove (come lo Unix System V) spesso non apprezzate dai "monopolisti". Tale contrapposizione nei fatti è oggi completamente tramontata in quanto gli Open Standard, de iure o de facto, sono universalmente accettati da tutti e sono introdotti su tutti i prodotti di mercato. Occorre solo porre attenzione ai falsi standard, ovvero a quei prodotti erroneamente ritenuti standard ma nei fatti solo molto diffusi e spesso non basati neppure su Open Standard de facto. Una scarsa attenzione a questo tema rischia di creare nuovi legami rigidi tra l'Applicazione e l'Operating Environment che rendono poi inutile o impossibile ogni processo di Selezione, consentendo la nascita nel mercato di nuovi "monopolisti".

Prodotti proprietari, efficienti, manutenuti, con un ottimo rapporto costo/prestazioni, purché basati su Open Standard (Java, TCP/IP, XML, ...), rappresentano spesso la migliore soluzione per la realizzazione dello stack e per lo sviluppo della Applicazione e consentono un corretto processo di scelta dell'Operating Environment basato sui Requisiti Non Funzionali.

4.0.3 Requisiti Non Funzionali

Definiamo caratteristiche o **Requisiti Funzionali** di una Applicazione Informatica tutto ciò che è richiesto relativamente al suo funzionamento, ossia tutto ciò che è necessario affinchè l'applicazione faccia le cose per le quali è stata pensata e realizzata, ovvero funzioni correttamente. Ad esempio, se una Applicazione Informatica deve stampare ogni giorno le fatture prodotte da una azienda per essere spedite ai clienti, un criterio funzionale per l'Applicazione è che sia in grado di

[5] In effetti il Sistema Operativo Linux che si appoggia sulle varie Architetture Hardware è differente nel nucleo ed in alcuni device driver, tuttavia il livello di portabilità della applicazione è totale a causa delle caratteristiche di progetto di Linux.

"stampare", la capacità di stampare è quindi un requisito funzionale senza il quale l'applicazione di Stampa delle fatture non può dirsi funzionante.

Riteniamo che i criteri funzionali non possano essere negoziati, in quanto essi sono un requisito per il funzionamento dell'Applicazione, quindi tutte le Piattaforme Informatiche che non sono in grado di rilasciare quanto richiesto dai requisiti funzionali devono essere escluse dal processo di Selezione: non ha senso infatti scegliere o prendere in esame Piattaforme sulle quali l'applicazione non può funzionare; ad esempio, se la nostra applicazione richiede funzioni di grafica avanzata per elaborare immagini, non ha senso selezionare per essa Sistemi Operativi che non sono in grado di supportare la grafica.

Definiamo caratteristiche o **Requisiti Non Funzionali** di una Applicazione Informatica tutti quegli elementi che non sono richiesti esplicitamente per il funzionamento della stessa, ma possono solo variarne alcuni aspetti, rendendola più veloce, meno costosa o più gestibile, in poche parole migliorarne o peggiorarne il funzionamento.

Sono caratteristiche o Requisiti Non Funzionali ad esempio:

- Rapporto Costo/Prestazioni, cioè le prestazioni della Piattaforma in relazione al suo costo, ovvero il suo "**costo specifico"**. Il tema della valutazione corretta dei costi è materia estremamente delicata che sarà trattato in dettaglio più avanti, ricordiamo solo che tale valutazione passa dalla definizione e determinazione di "Sistemi Equivalenti" su diverse Piattaforme Informatiche. Senza tale definizione non è possibile determinare una "prestazione equivalente" e di conseguenza una differenza tra i costi specifici. Stiamo considerando il rapporto costo/prestazioni piuttosto che la prestazione in se, poiché riteniamo che la "prestazione adeguata" debba ritenersi un requisito funzionale[6] e che quindi le Piattaforme Informatiche che non fossero in grado di fornire prestazioni adeguate in una qualsiasi configurazione debbano essere escluse dalla selezione.
- **Costi di Esercizio/Gestione**, si tratta di un tema importante, spesso fondamentale: i costi di esercizio sono quelli ricorrenti ovvero continuativi. La ottimizzazione dei costi ricorrenti è un tema molto caro specialmente alle Aziende quotate in borsa o agli Enti Pubblici. La riduzione di tali costi è uno dei driver fondamentali per la valutazione. Il tema dei costi di esercizio è particolarmente legato all' Operating Environment e per questo è relativo alla Piattaforma; la maggiore o minore "ricchezza" dell'Operating Environment in termini di supporti alla gestione (utilities) può rappresentare un maggiore o minore costo di esercizio.
- **Costi di Acquisizione,** nel caso di nuova infrastruttura sono i costi degli apparati e dei prodotti da acquisire, nel caso di trasformazione da una piattaforma ad un'altra sono i costi della trasformazione, compresi nuovi apparati o nuovi prodotti da acquisire.

[6] E' uso comune ritenere che qualunque sia la dimensione della Applicazione Informatica o il volume dei dati da trattare, esista sempre una configurazione di Sistemi per ciascuna Architettura Informatica in grado di fornire un livello di prestazioni adeguate per l'Applicazione: la pratica ci dimostra che ciò non è sempre vero.

- **Sicurezza**, ciascuna Piattaforma possiede intrinseche caratteristiche di Sicurezza (o di Vulnerabilità). La Sicurezza del Sistema e della infrastruttura nel complesso rappresenta non solo un rischio di impresa, ma anche un costo da sostenere in Strumenti ed in organizzazione.
- Caratteristiche di **Gestibilità**, sono quelle fornite complessivamente dall'Operating Environment, si traducono in efficienza (o inefficienza) ed in maggiori costi da sostenere per acquisire strumenti di gestione o personale.
- **Continuità** di Servizio, ovvero la capacità della Piattaforma di realizzare sistemi weakly coupled (detti comunemente cluster) in grado di assicurare continuità di Servizio a fronte di guasti o cadute di una parte della infrastruttura. Il tema della Continuità di Servizio è particolarmente sentito in strutture che necessitano una attività continua nelle 24 ore (ad esempio per servizi forniti sulla rete).
- Reattività alle variazioni di carico, meglio nota come **Resilienza**, assume importanza in tutti quei casi in cui il carico di lavoro da eseguire non è noto a priori, come ad esempio le applicazioni che danno servizi in rete con numero di utenti molto variabili nel tempo; si estrinseca nella capacità dell'Hardware di mettere a disposizione all'occorrenza nuove risorse[7] e dei Sistemi Operativi di utilizzarle[8].
- Capacità di **Disaster Recovery,** anch'essa può essere una caratteristica della Piattaforma Informatica che deve fornire strumenti per la realizzazione di cluster di sistemi remoti e per la copia sincrona dei dati tra due locazioni distanti (Data Replication).

Le caratteristiche Non Funzionali sono la base per il processo di Platform Selection. Spesso si basa la selezione solo sui Costi Totali della Piattaforma (modello TCO), in questa modo si tende a ricondurre ciascuna caratteristica Non Funzionale ad un costo (o ad un minor costo). Tale processo non è immediato nella maggioranza dei casi, pertanto, in prima approssimazione, i modelli basati sui costi cercano di utilizzare i Costi certi, cioè a valorizzare le componenti non funzionali immediatamente riconducibili ad un costo (Costo Specifico, Costi di Gestione, etc.) ed a considerare solo qualitativamente le altre componenti di costo.

Questo approccio però potrebbe essere fuorviante, ovvero potrebbe produrre una valutazione errata di alcune delle piattaforme considerate. E' certo che le diverse componenti Non Funzionali potrebbero avere peso diverso in relazione al tipo di lavoro o al tipo di Azienda/Ente. Sarà quindi opportuno stabilire all'inizio di ciascun processo verso quali elementi ci si vuole focalizzare, definendo come in ogni processo di ottimizzazione o selezione dei criteri o driver.

4.0.4 I driver della Platform selection

Fra i requisiti non funzionali che sono anche driver della scelta avremo, ad esempio, le prestazioni, le caratteristiche di gestibilità operative, i costi di esercizio, ecc.

[7] Tali risorse devono essere comunque presenti ed attivabili all'occorrenza, senza creare interruzioni nelle attività corso sul Sistema.

[8] La capacità dei Sistemi Operativi di usare o di rilasciare risorse è una caratteristica pregiata, essa fornisce alla Applicazione la possibilità di adeguarsi alle variazioni di carico senza interruzioni di servizio.

Ricordiamo tuttavia che nei processi valutativi della Piattaforma Informatica assumono spesso valore anche criteri diversi da quelli sopra elencati, ad esempio una Azienda/Ente potrà decidere di eseguire una Applicazione su una data Piattaforma indipendentemente dalla valutazione di criteri non funzionali, basandosi invece sulla presenza in azienda di molte altre applicazioni sulla stessa Piattaforma e ciò allo scopo di ridurre il numero di Piattaforme diverse presenti in Azienda. La Platform Selection quindi può e probabilmente deve essere mediata da valutazioni Strategiche ed Organizzative.

Torniamo al modello basato sulle caratteristiche Non Funzionali più importanti e quindi determinanti ai fini della Platform Selection, tra queste possiamo annoverare:

1. i costi di esercizio dell'Applicazione in relazione all'Operating Environment analizzato;
2. la presenza in Azienda/Ente di personale con esperienza specifica (skill) su quella data Piattaforma;
3. le prestazioni specifiche della Applicazione su quella piattaforma;
4. i Costi di Acquisizione di Hardware e Software.

Il processo di Platform Selection viene eseguito per applicazioni portabili; viene spesso eseguito "a posteriori" su Applicazioni già in uso, con lo scopo di ottimizzarne l'Operating Environment.

Ciò non toglie che tale processo possa essere eseguito "a priori", ovvero in fase di progettazione e rilascio di una nuova Applicazione. In questo caso esso assumerà il ruolo di analisi preventiva, allo scopo di determinare "a priori" quale possa essere l'Operating Environment più opportuno per eseguire l'applicazione, una volta completata. Sempre più spesso questo modo di operare "a priori" si sta diffondendo presso gli sviluppatori ed i progettisti, fornendo tra l'altro l'opportunità di selezionare gli strumenti di sviluppo ed i Middleware, scegliendo quelli che presentano caratteristiche di portabilità e standard, rispetto a quelli che sono molto diffusi.

Ci aspettiamo quindi nei prossimi anni la nascita di Applicazioni sempre più portabili, le quali fanno uso sempre più esteso di open standard con un incremento dei processi di Platform Selection sia "a priori" che "a posteriori".

Spesso, preso atto della non portabilità di una data Applicazione e comunque nell'intento di migliorarne l'Operating Environment in base a criteri definiti (ad esempio ridurne i costi), vengono valutati ed avviati processi anche molto complessi di trasformazione volti a ridefinire o riscrivere l'Applicazione usando strumenti che ne garantiscano la portabilità. Tali processi prendono in generale il nome di "Legacy Transformation".

Non è detto che un processo di migrazione su vasta scala, volto a riportare una Applicazione su standard di portabilià, risulti comunque conveniente ed opportuno: per questo è necessario disporre di metodologie e criteri valutativi di tipo oggettivo. Anche sul tema della Legacy Transformation ci aspettiamo grandi interventi negli anni a venire, per tale ragione e allo scopo di valutare tali processi in maniera quantitativa e asettica, ci servono strumenti di valutazione oggettiva del processo.

Nel caso in cui un'applicazione esistente e portabile venga trasportata su una diversa Piattaforma Informatica solo in base a criteri non funzionali (tipicamente una riduzione dei costi operativi) si suole parlare di processo di "re-hosting". Il **re-hosting** è quindi un processo di porting di una Applicazione da una Piattaforma a un'altra allo scopo di migliorarne alcuni aspetti non funzionali.

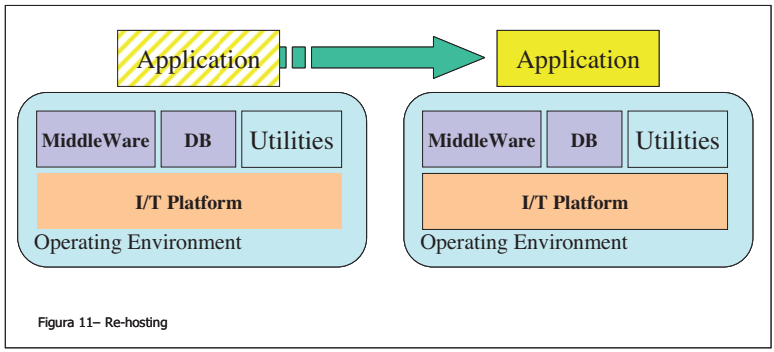


Figura 11 Re Hosting

Nella realtà i processi di re-hosting tendono ad essere eseguiti minimizzando l'impatto sul codice: ciò può essere reso possibile grazie ad opportuni strumenti o alla presenza di Middleware. La valutazione di un processo di re-hosting è materia sufficientemente complessa; spesso questo processo viene proposto semplicemente sulla base di una evidente riduzione dei Costi Operativi di gestione dell'Operating Environment, senza dare molto peso agli aspetti di gestione dell'ambiente finale o ai costi di acquisizione e migrazione. E' necessario osservare che una operazione di re-hosting rappresenta sempre lo spostamento della Applicazione tra Operating Environment differenti (vedere Figura 11). Ciò comporta in generale una variazione nel modo di operare che deve essere tenuta nella giusta considerazione e valutata opportunamente. Gli elementi che governano il processo di Selezione, in particolare le caratteristiche non funzionali, possono essere considerati con diverso peso, tendendo a trascurarne alcuni. Allo scopo di evitare valutazioni "emozionali" e non accurate occorre mettere a punto:

1. una Metodologia di Analisi che fornisca i Parametri in base ai quali valutare i Requisiti Non Funzionali, ovvero definire quali siano tra essi quelli maggiormente rilevanti;
2. un modello rigoroso di Analisi dei dati forniti dalla metodologia, in grado di dare un valore a ciascun elemento non funzionale;
3. un criterio preciso ed il più possibile oggettivo per stimare le potenze degli elaboratori nelle diverse piattaforme (sistemi equivalenti);
4. una serie di altri criteri ottimali per valutare tutto ciò che non è numericamente quantificabile (Best Practices), ad esempio il numero di persone necessarie per gestire una data Piattaforma rispetto ad un'altra.

4.0.5 Il Metodo

Il metodo che vogliamo seguire è schematizzato nella Figura 12. Supporremo di voler utilizzare dal punto di vista quantitativo solo i Driver di costo riservando solo un ruolo qualitativo ad altri elementi come l'affidabilità, la sicurezza, l'Operating Environment, la presenza di skills pregressi in azienda, etc[9].

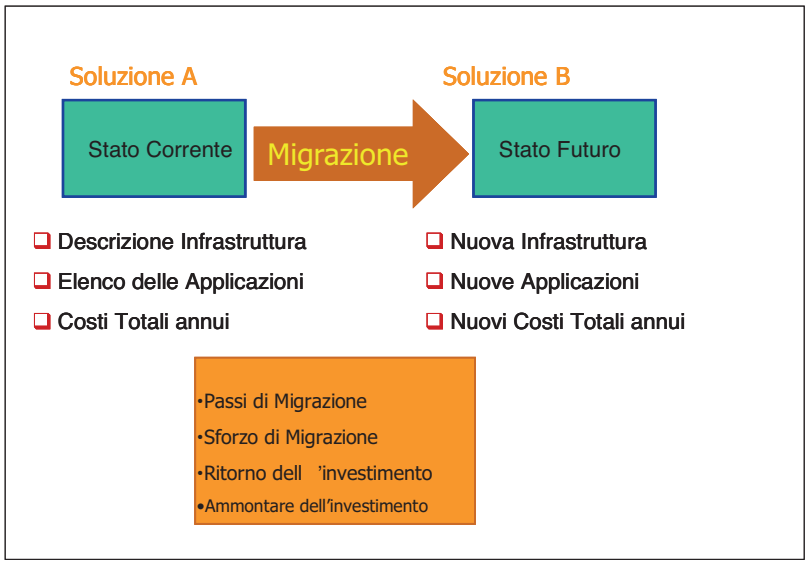


Figura 12 Il Metodo

Per rendere maggiormente accurato il calcolo sarà opportuno inserire solo i costi determinati con certezza, indicando con chiarezza se e quando si tratta di stime. Il metodo consiste nel confronto diretto tra due o più possibili soluzioni infrastrutturali (Piattaforme informatiche o aggregati di esse) in grado di eseguire (ospitare) la nostra (o le nostre) applicazioni informatiche oggetto di analisi.

E' bene ricordare che per complesso che sia l'ambiente operativo globale oggetto dell'analisi, esso sarà comunque riconducibile sempre ad un insieme discreto di Piattaforme che devono soddisfare il requisito primario della portabilità delle Applicazioni in esame. Se una o più non risultassero portabili in maniera immediata,

[9] Qualunque tipo di driver può essere ricondotto ad un costo (diminuizione o incremento); ogni interruzione di servizio è un costo, quindi il poter disporre di un Operanting Environment più efficiente si traduce nell'avere bisogno di minori risorse umane di tipo gestionale. In pratica solo poche aziende o enti sono in grado di valutare con precisone tali valori e, nel dubbio, risulterà opportuno inserire i soli elementi certi nel calcolo, indicando in modo qualitativo i rimanenti.

sarà necessario stimare lo sforzo (effort) di migrazione necessario ed ovviamente ricordurlo ad un Costo che defineremo "Costo di Migrazione"[10].

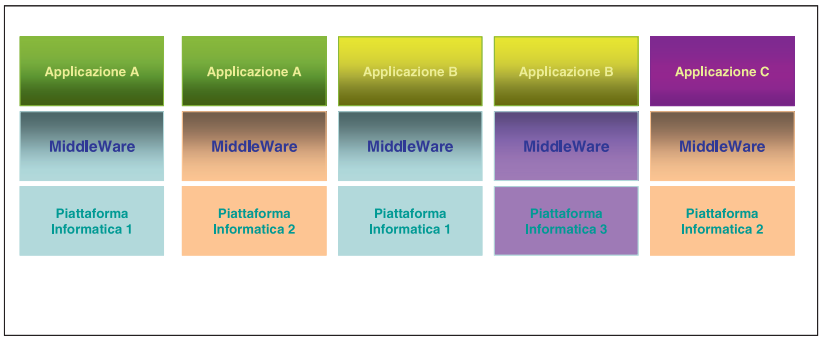


Figura 13 Infrastruttura composta da un insieme di Piattaforme

Il problema iniziale si riconduce a:

1. elencare le componenti atomiche della Infrastruttura Informatica in esame secondo lo schema della Figura 13;
2. definire per ciascuna un costo iniziale e un costo operativo;
3. valutare una alternativa per ciascun elemento modificando[11] la Piattaforma in base al concetto di Portabilità;
4. valutare per ciascuna Funzione di Servente Logico (cioè per ciascun blocco applicativo) un Sistema Equivalente secondo la nuova Piattaforma prescelta[12];
5. stimare lo sforzo ed il tempo di migrazione;
6. calcolare il costo della soluzione alternativa (valori e andamento nel tempo);
7. calcolare il periodo di Ritorno dell'Investimento (ROI).

Il risultato di una tale metodologia si esprime in uno o più valori di costo dei quali assumeranno particolare rilevanza le seguenti:

- una rappresentazione comparativa delle due o più soluzioni che indichi la più conveniente;
- un grafico di tipo temporale che indichi il punto (periodo) di Ritorno dell'investimento (ROI);
- un elenco degli elementi qualitativi che non è stato possibile o non si è voluto esprimere in modo quantitativo.

[10] Un Costo di Migrazione, anche se minimo, è sempre presente quando si trasporta una Applicazione Informatica da una piattaforma ad un'altra: basti pensare al costo e ai tempi necessari per la copia fisica dei programmi o per l'approntamento della nuova infrastruttura.
[11] Alcune Funzioni di Servente potranno restare sulla stessa piattaforma.
[12] L'Equivalenza dei Sistemi è uno dei cardini su cui si basa il processo.

Attraverso l'esame delle figure di costo ottenute, la valutazione del periodo (Tempo di ritorno dell'investimento) e la valutazione degli elementi qualitativi non inclusi nel calcolo si potrà operare un confronto ed una classifica delle possibili alternative scegliendo (selezionando) la più adatta.[13] I processi di Selezione non sono mai rigidi, crediamo tuttavia che sia una buona approssimazione iniziale una valutazione basata solo sui costi o sul ricondurre tutti gli elementi ad un costo; se non altro ciò consentirà di evitare grossolane sviste sin dalle prime valutazioni o di raffreddare ingiustificati entusiasmi prima di avviare un processo di porting.

E' chiaro che tutto il processo ha due elementi la cui determinazione risulta di vitale importanza ai fini della selezione e cioè:

1. la capacità di determinare con precisione un Sistema Equivalente per ogni Funzione di Servente per tutte le Piattaforme in esame;
2. la capacità di valutare il grado di portabilità della Applicazione attraverso le Piattaforme considerate e di stimarne con precisione lo sforzo di Migrazione in termini di tempo[14] e risorse necessarie.

4.0.6 Il Valore della Tecnologia

Descrivendo il processo di selezione si deve tenere conto di alcuni aspetti prettamente tecnologici ed architetturali che risultano essere essenziali nel corso del processo stesso in quanto abilitanti rispetto ad un percorso di ottimizzazione. Ciascuna Piattaforma presenta caratteristiche peculiari e si differenzia dalle altre in termini prettamente tecnologici. E' noto ad esempio come le piattaforme basate sul Mainframe (z/OS + zArchitecture ovvero Linux + zArchitecture) siano particolarmente efficaci nel trattamento di grandi volumi di dati acceduti contemporaneamente da un gran numero di utenti, laddove le piattaforme di tipo UNIX (AIX + Power Risc ovvero Linux + Power Risc) si presentano particolarmente adatte ad operazioni di calcolo intensivo e così via.
Sarà quindi opportuno assegnare un valore all'interno del processo di selezione anche agli aspetti tecnologici della piattaforma.
Tale procedimento non è semplice: ciascuna piattaforma nel tempo tende progressivamente a migliorare gli aspetti di eventuale debolezza[15]. Perciò volendo muoversi su basi quantitative certe, sarà opportuno considerare le differenze

[13] E' costume di molti consulenti produrre una classifica in base ad ognuno dei driver (ad esempio Costo, Sicurezza , Esercibilità, etc.). Questo metodo non è del tutto efficace in quanto lascia spazio ad elementi non rigorosi o non rigorosamente determinati.
[14] Il Tempo di migrazione è un elemento di cui occorrerà tenere conto, poiché sarà necessario mantenere le due infrastrutture contemporeneamente attive per tale periodo di tempo.
[15] Ad esempio la zArchitecture ha annunciato nel 2008 l'introduzione di una particolare assist Hardware, denominata Decimal Floating Point Accelerator, in grado di accelerare le operazioni in virgola mobile fino a dieci volte; ciò ha posto la zArchitecture in una nuova luce rispetto al calcolo numerico intensivo, analogamente le capacità di I/O dei Sistemi Power RISC migliorano progressivamente negli anni, il che rende il paradigma della specializzaizone dei Sistemi sempre meno verificato.

architetturali tra le piattaforme solo in termini prestazionali[16], ovvero assumeremo che esiste sempre una possibile configurazione in grado di soddisfare tutti i requisiti di un qalunque progetto. Tale affermazione non è sempre vera, infatti potrebbero verificarsi dei casi per i quali alcune piattaforme non risultino in grado di soddisfare i requisiti prestazionali richiesti (in questi casi e solo in questi sarà corretto escludere alcune piattaforme "ab initio", in quanto non conformi al requisito prestazionale a priori).

Per tutti gli altri casi varrà l'approssimazione sopra descritta e cioè si assumerà che la piattaforma sia in grado di eseguire il lavoro richiesto con prestazioni accettabili con una configurazione determinata, comunque grande.

Altri aspetti prettamente tecnologici di cui tenere conto sono quelli legati all'efficienza della piattaforma stessa, tra questi è utile considerare:

1. le Dimensioni Massime di un Singolo Sistema: esse entrano in gioco sul tema della scalabilità verticale, cioè del massimo carico di lavoro che può essere ospitato su una singola funzione di servente; esso ovviamente dipende dalla capacità massima del singolo servente fisico. Più potente è la singola macchina, minore sarà il numero di macchine fisiche necessarie e sufficienti ad eseguire un dato lavoro;
2. le Capacità del Sistema Operativo di eseguire Workload Eterogenei simultaneamente sulla stessa Istanza di Sistema: ciò consente una sostanziale riduzione del numero di serventi logici (funzione di servente); la riduzione del numero di Serventi logici rappresenta un fattore di consistente risparmio nei costi operativi;
3. le capacità di realizzare dei Cluster di Sistemi: tale funzione è principalmente legata alla continuità operativa, ovvero alla capacità di continuare le operazioni a fronte di un guasto o di indisponibilità di parte delle apparecchiature. I cluster possono essere realizzati a distanze piccole (Campus), a distanze metropolitane o possono essere geograficamente dispersi;
4. le capacità di Virtualizzazione e di Resource Sharing della Piattaforma: è la tecnologia più importante ai fini del consolidamento fisico delle funzioni di servente su un numero inferiore di macchine fisiche. La virtualizzazione consente di svolgere lo stesso carico a parità di serventi logici su un numero inferiore di Sistemi Fisici senza alcuna modifica al Software dei Serventi logici, con evidente risparmio di costi operativi e migliori capacità di realizzare Sistemi per il Disaster Recovery. Le capacità di virtualizzazione sono differenti sulle varie piattaforme. La figura 5 mostra un Sistema virtualizzato: ad oggi i Sistemi virtualizzati consentono di attivare diversi Sistemi Operativi su una stessa architettura hardware; sono quindi in grado di ospitare diverse piattaforme sullo stesso Sistema Fisico[17]. Fattore differenziante tra le diverse realizzazioni di un Sistema Virtualizzato è rappresentato dalla tecnica usata per la virtualizzazione e dalla capacità di condivisione di risorse: avremo quindi virtualizzatori basati su Hardware

[16] Nel prossimo capitolo congloberemo queste differenze in un valore denominato WLF Workload Factor.

[17] In futuro sarà possibile disporre di virtualizzatori IBRIDI, cioè in grado di ospitare diverse architeturre Hardware + Diversi Sistemi Operativi, tra i quali alcuni di tipo Ibrido.

(Hypervisors) o su Software. Alcuni di essi saranno capaci di condividere la stessa risorsa fisica (ad esempio il processore) tra diverse Funzioni di servente logico, ovvero necessiteranno di risorse dedicate.

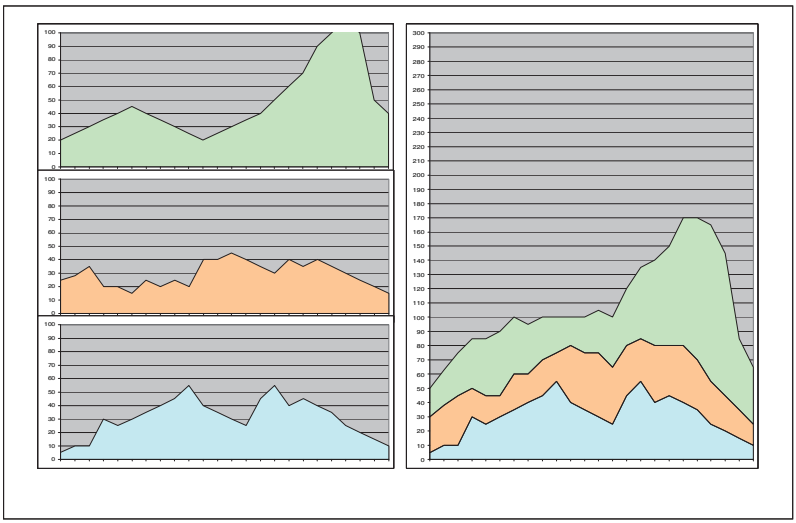


Figura 14 Condivisione = Risparmio di Risorse

Ciascuna delle caratteristiche descritte è diversamente quantificabile, ad esempio la Virtualizzazione rappresenta un forte abilitatore in termini di efficienza infrastrutturale, in quanto consente di ridurre il numero di Macchine fisiche necessarie per realizzare la singola funzione di Servente Logico; tutto ciò si può facilmente ricondurre ad un costo inferiore in termini di gestione; al contrario, la capacità di realizzare dei Cluster di Sistemi attiene al concetto di continuità operativa (Business Continuity) che non è facilmente quantificabile in termini di costo. In ciascun caso quindi sarà opportuno decidere quali elementi Tecnologici potranno essere inclusi nel processo di selezione come 'facilitatori' (enablers) e quali di essi potranno essere facilmente quantificati.

Un discorso a parte si deve fare a proposito del Sistema Operativo Linux, molto utilizzato nei processi di "**Cambio di Piattaforma**" (o re-hosting). Si dice che si esegue un processo di re-platforming (o re-hosting) quando una o più funzioni del Servente vengono spostate da una Piattaforma ad un'altra. Tale processo ha senso solo quando la piattaforma di arrivo è ritenuta maggiormente conveniente, ad esempio dal punto di vista dei costi, ovvero più affidabile, ovvero più utile al processo stesso di ottimizzazione. Temi fondamentali del processo di re-platforming sono:

- la **compatibilità applicativa**, ovvero la capacità della piattaforma di arrivo di ospitare la stessa funzione applicativa oggetto di re-platforming

(re-hosting) con le stesse caratteristiche fuzionali. Ciò dipende dalla portabilità della funzione o della Applicazione Informatica;
- la **Pianificazione della Capacità** (Capacity Planning), ovvero la determinazione della potenza del Sistema ricevente che, essendo in generale di un'altra architettura, potrà risultare caratterizzato da metriche diverse.

Un caso particolare di **compatibilità applicativa** e di conseguente portabilità si ha per le cosidette Funzioni Infrastrutturali o di Servizio, quali ad esempio quelle che gestiscono la Sicurezza, le autenticazioni, le funzioni di Stampa, di File Serving, di Directory, di Name Serving, ecc. Tutte queste funzioni sono praticamente analoghe su tutte le piattaforme e pertanto sono automaticamente portabili. Ne segue che in relazione ad esse la portabilità è genericamente garantita e quindi il re-platforming (re-hosting) è quasi sempre possibile. Analoga osservazione può essere fatta tutte le volte in cui l'Applicazione usi dei Sottosistemi o dei Middleware; in questo caso la portabilità e quindi la capacità di essere ospitata sull'altra piattaforma è garantita dal Middleware. Non si ha invece portabilità e quindi non si può cambiare facilmente la Piattaforma quando si usano strumenti e linguaggi proprietari o tipici della Architettura o Piattaforma, come Assembler, o se si usano Gestori di Basi Dati che funzionano su una sola piattaforma (ad esempio SQL Server).

Nel processo di cambio di Piattaforma assume una particolare importanza il Sistema Operativo Linux al fine della ottimizzazione di una infrastruttura. E' noto a tutti infatti come Linux possa ospitare, a costi di Esercizio e di Acquisizione più contenuti, elementi dell'Infrastruttura Informatica come i già citati DNS, LDAP Server, File Server, ecc. Inoltre, per le caratteristiche di Open Source, sempre più spesso funzioni Applicative importanti vengono ospitate sul Sistema Operativo Linux, risultando nel complesso più gestibili e più semplici da manutenere.

Va comunque precisato che Linux non è in grado di svolgere con la stessa qualità di Servizio funzioni complesse, ora svolte da Sistemi Operativi più potenti e completi come z/OS, e che quindi al momento non è pensabile che esso possa completamente sostituirli. Questa osservazione spinge ad affermare che all'interno del processo di selezione occorrerà tenere conto anche della possibilità di spostare alcune funzioni da piattaforme più costose a Linux, ma ci si dovrà limitare a quelle che non richiedono la qualità di Servizio di Sistemi Operativi più complessi, facendo leva sulle capacità di virtualizzazione offerte dalla architettura e dalla piattaforma.

4.0.7 Confronto tra Sistemi Eterogenei

Il processo che consente di determinare il confronto tra potenze elaborative di due o più Sistemi di diversa Piattaforma prende il nome di "Cross Platform Sizing" (Dimensionamento multipiattaforma). Ha senso porsi tale problema tutte le volte in cui si vogliano fare confronti tra le Piattaforme; il metodo illustrato in figura 12 ad esempio impone tale processo per stabilire quale sia l'Infrastruttura di Serventi da utilizzare per eseguire un certo carico di lavoro (Applicazione) quando vogliamo considerare due infrastrutture alternative.

Il problema si traduce nello stabilire metriche equivalenti o universali in grado di valutare le capacità elaborative, indipendentemente dalla Piattaforma Informatica. In

assenza di standard internazionali per misurare la capacità delle Piattaforme Informatiche il tema risulta a tutt'oggi non risolto nella sua interezza; infatti, mentre alcuni organismi autonomi eseguono misurazioni comparative delle piattaforme basate su UNIX costantemente (tali misurazioni comparative vengono regolarmente pubblicate), ciò accade meno per Linux e non accade per i Sistemi di zArchitecture, per i quali IBM ha sviluppato un Sistema proprietario denominato Large System Performances Report (LSPR), che mette a confronto i vari modelli di macchina con vari sistemi operativi (ad esempio z/OS, z/VM e zLinux). Nei fatti non esistono misurazioni pubblicate che mettano in relazione la piattaforma zArchitecture + z/OS con PowerRisc + AIX (solo per restare in ambito di prodotti IBM).

Ulteriore complicazione è data da due altre considerazioni:

- alcune piattaforme informatiche basate sullo stesso Hardware (ad esempio x86) con Sistemi Operativi diversi (ad esempio MS Windows o Linux) presentano livelli di efficienza radicalmente diversi e quindi valori di potenza differenti per lo svolgimento delle stesse funzioni applicative;
- la stessa piattaforma presenta livelli di efficienza (e quindi potenza relativa) diversi nell'eseguire lavori di diversa natura (ad esempio Batch piuttosto che Transazionali).

Esiste infine una sostanziale differenza di efficienza (in parte dovuta a questioni puramente architetturali, in parte alla diversa "esperienza" dei Sistemi Operativi) tra le capacità di potenza del Sistema Operativo z/OS e quelle di Linux sulla stessa architettura hardware. Proponiamo due possibili metodi per determinare l'equivalenza dei sistemi di elaborazione, essi sono:

1. il **metodo del Sistema Equivalente:** consiste nel determinare l'equivalenza dei Sistemi solo basandosi su parametri tecnici prestazionali secondo un opportuno fattore di conversione tra le architetture;
2. il **metodo del Lavoro Equivalente:** consiste nel determinare l'equivalenza dei Sistemi in base al fatto che essi siano in grado di svolgere lo stesso carico di lavoro (comunque determinato) con le stesse prestazioni.

I due metodi proposti sono sostanzialmente differenti nella forma e spesso conducono a risultati diversi, il primo in particolare tende ad una sottostima del sistema 'non noto', ciò a causa della difficoltà di stabilire delle metriche di confronto tra Sistemi di architettura diversa; il secondo invece è più accurato ma necessita di complesse misure e comunque di disporre di entrambi i Sistemi da confrontare. Questo ultimo metodo è tipico dei "Pacchetti Applicativi" o package prodotti dalle Società di Software (Software House): in questo caso è cura del fornitore del package dichiarare quali sono i Sistemi di ciascuna piattaforma in grado di svolgere un dato volume di lavoro; il fornitore fa le misurazioni e pubblica i risultati.

Entrambi i metodi proposti sono sempre soggetti a verifica, infatti l'unico vero modo di stabilire una equivalenza è quello di misurare le prestazioni di un lavoro noto sui sistemi da confrontare; questa tecnica, che è l'unica precisa, prende il nome di "benchmarking". Un benchmark, se pubblicato, rappresenta una certezza sperimentale di prestazione su un carico di lavoro noto.

Purtroppo non e' sempre possibile effettuare dei benchmark[18] ed è quindi necessario procedere mediante approssimazioni. Ove possibile, il metodo del lavoro equivalente è sempre il migliore, tuttavia, per varie ragioni, non è sempre possibile una sua applicazione puntuale. Vediamo quindi l'altro metodo, ovvero quello del Sistema Equivalente, che in molti casi è l'unico applicabile in pratica[19].

Premettiamo che in generale le prestazioni di un Sistema non dipendono solamente dalla capacità elaborativa del Calcolatore; l'architettura hardware ha un ruolo essenziale, ad esempio nel trattamento delle Operazioni di I/O o nell'uso dei dispositivi e della memoria. Inoltre il Sistema Operativo può presentare caratteristiche di progetto che lo rendano più adatto ad un certo tipo di operazioni e che ne penalizzino altre. Avremo così ad esempio:

- Sistemi specializzati per il calcolo intensivo
- Sistemi specializzati per l'elaborazione dei dati
- Sistemi specializzati per operazioni commerciali (transazionali)
- Sistemi specializzati per il disegno e la grafica
- .....

Premesso quindi che le prestazioni dipendono sia dalla architettura hardware che dal Sistema Operativo (cioè dalla Piattaforma) ed assodato che diversi tipi di lavoro possono avere prestazioni diverse a parità di Piattaforma, sarà opportuno valutare le capacità elaborative, e quindi le prestazioni, precisando anche il tipo di lavoro eseguito; definiremo questa caratteristica **Workload Factor** - WLF.

E' noto che i Sistemi non riescono sempre a lavorare al massimo della potenza che sono in grado di fornire (sulla carta). Per esempio, sappiamo dalla pratica che alcune Piattaforme sono tipicamente utilizzabili in modo efficiente per percentuali del 60% al massimo, mentre altre si spingono vicino al 95%. Questo non è un errore di progetto, ma una caratteristica globale e peculiare del Sistema, legata in buona parte all'architettura Hardware ed in altra parte al disegno del Sistema Operativo. Questa caratteristica definisce un Punto di Saturazione del Sistema detto **System Saturation Point** (SSP). I Sistemi centrali Mainframe presentano un SSP circa uguale ad uno (100%), alcuni Sistemi di architettura Hardware x86 non superano il 60%.

Infine non esiste uno standard internazionale per definire le metriche relative alla potenza di un Sistema; il mercato mette a disposizione dei benchmark, generalmente correlati al tipo di lavoro svolto, oltre a misure di grandezze fisiche come il ciclo base dei Processori o i MIPS. E' opportuno osservare che per questo motivo i confronti tra Piattaforme vengono inficiati di una ulteriore imprecisione dovuta alla necessità di dover normalizzare le metriche. Nell'effettuare il confronto e quindi il dimensionamento dovremo precisare:

1. Architettura Hardware

[18] In effetti i benchmark pubblicati coprono una piccola parte dei carichi di lavoro possibili e coprono solo alcuni Pacchetti Applicativi a larga diffusione.

[19] E' opportuno precisare che l'inaccuratezza del metodo di confronto tra i Sistemi potrebbe inficiare pesantemente qualunque confronto tra piattaforme informatiche, sarà quindi opportuno, appena possibile, effettuare delle verifiche dei risultati ottenuti, ad esempio realizzando un prototipo che consenta di valutare la bontà delle stime operate.

2. Sistema Operativo
3. Tipo di Lavoro
4. Punto di Saturazione del Sistema
5. Metrica usata

Proviamo ad esprimere in forma quantitativa i concetti fino ad ora espressi, nella convinzione che comunque sia sempre necessaria una verifica sperimentale di quanto stiamo affermando. Baseremo il nostro metodo sulla seguente definizione di Sistemi Equivalenti.

Dati i **Sistemi** Informatici A e B, rispettivamente di capacità Ca e Cb, rispettivamente utilizzati Ua ed Ub, essi si diranno **equivalenti** quando:

Ca * Ua * Ka = Cb *Ub * Kb

In questo caso equivalente vuole dire che essi sono capaci di fare lo stesso lavoro nello stesso tempo con gli stessi risultati.

Ca e Cb sono le capacità dei Sistemi, Ua e Ub sono numeri compresi tra 0 ed 1 che ne rappresentano l'utilizzo percentuale, Ca e Cb devono essere espressi entrambi nella stessa unità di misura, Ka e Kb sono funzioni numeriche che tengono conto delle differenze architetturali e del tipo di lavoro svolto.

Le funzioni K, per quanto detto a proposito di SSP, oltre ad essere variabili in funzione del workload (WKL), saranno variabili in funzione dell'utilizzo ovvero:

Kx = f(WKL , Ux)

e fissato il WKL avremo quindi:

Kx = f(Ux)

se poniamo Ua=1 (SSP=100%) per il Sistema A e:

WLF = Ka/Kb= f(Ua)/f(Ub)

potremo scrivere la relazione di partenza nella forma:

Ca = (Cb * Ub) / WLF

Se calcoliamo la nostra relazione con Ub uguale a SSP di B, potremo con questa relazione esprimere la capacità necessaria sul Sistema A per eseguire il lavoro che porta il Sistema B al System Saturation Point, cioè abbiamo realizzato una equivalenza al limite di utilizzo massimo tra A e B in funzione del solo WLF. La funzione WLF fornisce risultati espressi in unità di misura dimensionalmente uguali a Cb/Ca. Ub è adimensionale. Per dare un valore pratico al calcolo dobbiamo quindi essere in grado di fissare una coppia di valori di utilizzo per le piattaforme che vogliamo confrontare e misurare lo stesso Tipo di Lavoro su entrambe. Ad esempio 60% per il Sistema B e 100% per il Sistema A. Per la coppia di utilizzi individuata WLF può definirsi costante e ad esempio di valore $\mathbf{WLF_{ab(100/60),}}$ in questo caso e solo per la coppia di utilizzi considerati ed il tipo di lavoro in esame varrà la semplice relazione:

Ca = Cb * Ub / $\mathbf{WLF_{ab(100/60)}}$

nella quale compaiono tutti valori noti tranne Ca.

Mediante questa relazione sarà particolarmente semplice, a meno di definire una metrica equivalente, determinare la capacità Ca equivalente a Cb sulle diverse piattaforme Informatiche A e B. Il problema, inizialmente molto complesso, si riduce ora a determinare sperimentalmente i valori di WLF relativi al SSP per le coppie di Piattaforme che vogliamo confrontare e a definire una metrica equivalente tra le Piattaforme A e B. Una prima semplice applicazione del metodo, valida come prima approssimazione, si ha utilizzando la frequenza di Clock del processore per misurare le capacità della Piattaforma.

Il problema che vogliamo risolvere è il seguente:

Dato il Sistema A con n processori di frequenza $_A$ con un utilizzo U_A, quanti processori saranno necessari sul Sistema B, con frequenza di Clock $_B$ per eseguire lo stesso lavoro con un utilizzo U_B.

Avremo quindi:

m = (n * $_A$ * U_A) / ($_B$ * U_B * WLF)

dove WLF è la costante misurata nel range di utilizzo previsto per la coppia di Piattaforme Informatiche A e B.

I risultati ottenuti con un simile calcolo sono approssimati in quanto la frequenza di clock del processore non è un elemento esauriente per definire le prestazioni di un Sistema.

Recentissimamente sono stati introdotti criteri più complessi basati sulla equivalenza tra i MIPS e alcune grandezze utilizzate da altri benchmark pubblicati[20].

Abbiamo quindi nuove relazioni del tipo:

MIPS = (rPerf * 100 * U_A) / (U_B * WLF)

dove compare ancora WLF come definito precedentemente ed il valore rPerf che rappresenta il fattore di conversione MIPS/RPE, con una relazione del tipo:

rPerf = RPE_A / KRPE

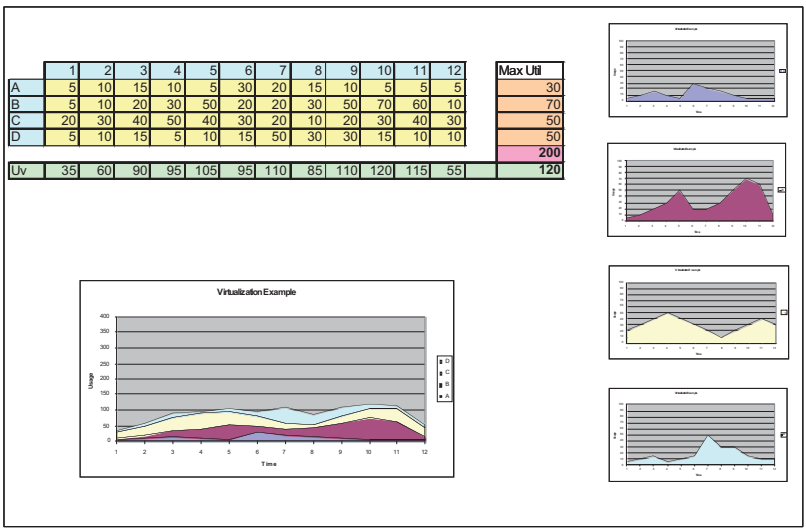
il valore KRPE non è di pubblico dominio.

Questa relazione, una volta conosciuto rPerf, consente di risolvere il problema del calcolo dei MIPS equivalenti ad un Sistema del quale si conosce l'RPE.

[20] RPE è un valore pubblicato dalla organizzazione Ideas International.

Poiché l'utilizzo U_X non è costante nel tempo, le relazioni qui introdotte sono tutte funzioni del tempo e quindi presentano un andamento variabile rispetto ad esso. Ai fini del nostro problema la dipendenza esplicita dal tempo non è importante, essendo conveniente nei processi di dimensionamento sempre riferirsi all'utilizzo massimo U_{max}. Non è opportuno infatti, come spesso si fa, riferirsi a valori medi su grandi periodi, bensì sul massimo calcolato su un intervallo sufficientemente piccolo rispetto ai tempi di risposta attesi dalla Applicazione Informatica. La pratica suggerisce tuttavia di procedere sempre a partire da valori misurati per uno dei due Sistemi per i quali determinare l'equivalenza.

La dipendenza dal tempo dell'utilizzo sarà invece molto importante nei processi di dimensionamento di Sistemi Virtualizzati: in questo caso la possibilità di condividere risorse tra vari utilizzatori (Sistemi Operativi) rende opportuno valutare la dipendenza dal tempo della funzione Utilizzo, mentre è sbagliato riferirsi ai soli massimi della stessa.



	1	2	3	4	5	6	7	8	9	10	11	12	Max Util
A	5	10	15	10	5	30	20	15	10	5	5	5	30
B	5	10	20	30	50	20	20	30	50	70	60	10	70
C	20	30	40	50	40	30	20	10	20	30	40	30	50
D	5	10	15	5	10	15	50	30	30	15	10	10	50
													200
Uv	35	60	90	95	105	95	110	85	110	120	115	55	**120**

Figura 15 Dimensionamento di Sistemi Virtualizzati

La figura 15 mostra, a titolo di esempio, l'utilizzo, nell'arco di una ipotetica giornata e per intervalli di tempo fissati arbitrariamente, di quattro Risorse Informatiche (ad esempio Server); la tabella in alto ne mostra i valori numerici. Come si può facilmente notare, essendo l'utilizzo variabile nel tempo con andamento differente, se si ipotizza di consolidare le quattro risorse su di una sola (condivisa) si osserveranno due importanti fenomeni:

1. il massimo utilizzo della risorsa condivisa (120) sarà sempre minore o uguale della somma dei massimi utilizzi di ogni singola risorsa (200);

2. Il punto (periodo della giornata) in cui si realizza il massimo utilizzo della risorsa condivisa potrebbe non coincidere con nessuno dei punti in cui si realizzano i massimi delle singole risorse.

Queste due importanti osservazioni sono state poste alla base del concetto di virtualizzazione ed usate come giustificazione della virtualizzazione stessa. Cerceremo di fornire un criterio quantitativo per il dimensionamento di processi di Consolidamento dei serventi: ciò vale sia nel caso di Serventi che usano la stessa Piattaforma, sia nel caso di Piattaforme eterogenee, fatto salvo quanto detto per la normalizzazione delle metriche.

Il problema del dimensionamento di Infrastrutture virtualizzate che consolidano molti Server diversi viene qui descritto al netto del carico aggiuntivo (overhead) dovuto al virtualizzatore, soprattutto quando esso è implementato mediante Software. In questo caso tale carico, generalmente noto, sarà semplicemente aggiunto al risultato finale sotto forma di percentuale o espresso in quantità di potenza di calcolo. L'esperienza ci dice che nei processi di dimensionamento di infrastrutture virtualizzate conviene partire sempre da dati di carico dei singoli serventi effettivamente **misurati** e non dedotti da stime o medie: ciò in quanto gli effetti delle medie su periodi lunghi potrebbero inficiare pesantemente il risultato.

E' necessario considerare per tale misura la scelta dell'intervallo di campionamento: osserviamo infatti la figura 16; essa mostra l'andamento in Ghz del "consumo" di quattro serventi fisici (che prevediamo di consolidare) e della loro somma (area blu), avendo misurato i valori medi ogni minuto.

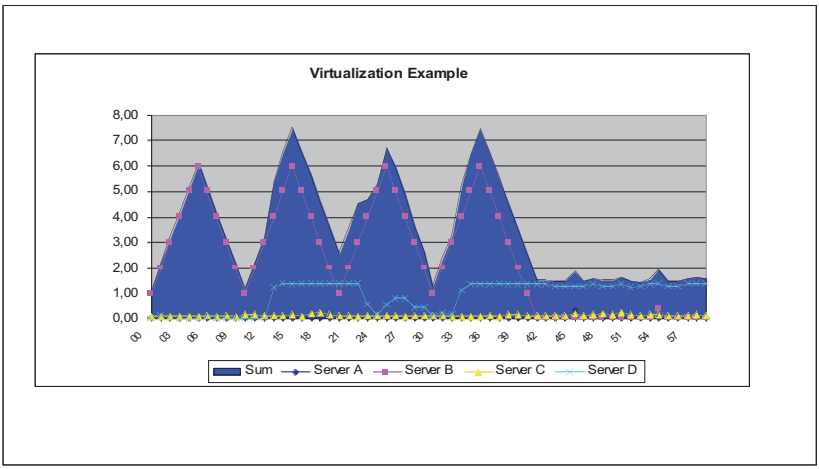


Figura 16 Intervallo di Campionamento

Dal grafico di figura 16 si evince che il consumo massimo, cioè la dimensione che dovrà avere il Sistema virtualizzato per poter contenere il carico dei quattro serventi divenuti logici, si trova intorno a 8 Gigahertz.

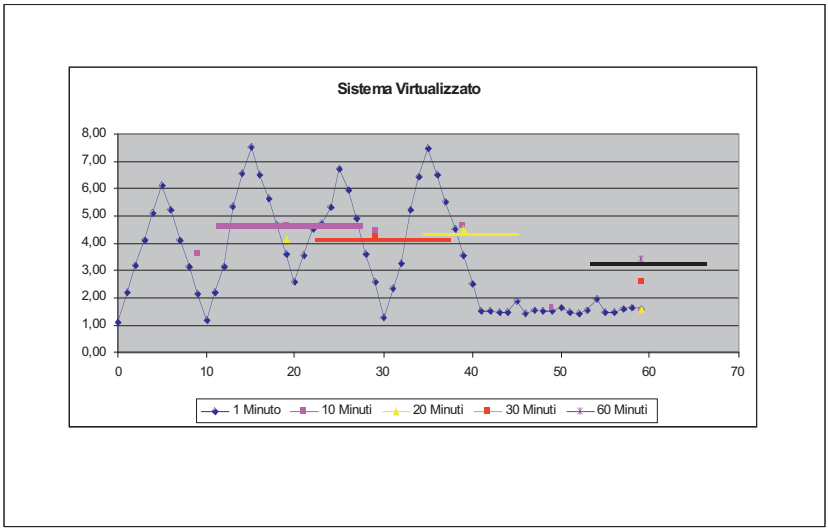


Figura 17 Dipendenza del Massimo dall'intervallo di Campionamento

La figura 17 mostra la dipendenza del valore massimo misurato dall'intervallo di campionamento: a partire dagli stessi dati (relativa ad una ora di lavoro), mostra le differenze nei valori massimi ottenuti calcolando le medie rispettivamente ogni 5, 10, 20 e 30 minuti (Barrette Continue). Come si vede, maggiore l'intervallo di calcolo della media (campionamento), minore il valore massimo; con l'intervallo di campionamento più alto (60 minuti), il consumo massimo aggregato si abbassa di 3 GHz. Il risultato pratico è che avremmo bisogno (il condizionale è d'obbligo) per consolidare di un server molto meno potente: eppure abbiamo esaminato lo stesso caso.

Ci chiediamo quindi quale possa essere un intervallo di calcolo della media ragionevolmente adeguato ai nostri scopi, anche perché potremmo osservare che riducendolo ulteriormente (cioè passando ad esempio da un minuto ad un secondo) rischieremo l'effetto inverso e cioè quello di percepire un consumo massimo sempre maggiore vanificando l'effetto della virtualizzazione: è facile infatti dimostrare che, considerando intervalli sempre più piccoli, ne esisterà almeno uno per il quale il consumo della risorsa sarà pari alla potenza complessiva disponibile (100%). Una ragionevole approssimazione può essere quella di considerare intervalli dello stesso ordine di grandezza della durata media di una transazione Applicativa (ad esempio minuti per le operazioni online con intervento umano ovvero ore per le transazioni di tipo Batch); è da considerare che il risultato di un sottodimensionamento sortirà l'effetto di prolungare i tempi di esecuzione.

Una volta fissato l'intervallo di calcolo delle medie e misurato l'utilizzo delle risorse che vogliamo virtualizzare per un periodo ragionevolmente rappresentativo (ad esempio un giorno, un mese, ecc.), ci ritroveremo per ogni servente una serie di misure del tipo:

$$U_x = [\, U_{x1}, U_{x2}, \ldots\ldots U_{xn}]$$

che rappresenta una serie di misure relative alla risorsa X per n Intervalli di tempo uguali. Se consideriamo **k** risorse diverse e se sincronizziamo il numero **n** e la durata degli intervalli, le nostre misure costituiranno una matrice rettangolare di **k** righe ed **n** colonne:

$$\begin{vmatrix} U_{11} & U_{12} & U_{13} & & U_{1n} \\ U_{21} & U_{22} & U_{23} & & U_{2n} \\ U_{31} & U_{12} & U_{33} & & U_{3n} \\ \ldots\ldots & & & & \\ U_{k1} & U_{k2} & U_{k3} & & U_{kn} \end{vmatrix}$$

La somma di ciascuna colonna rappresenta, per ogni intervallo di tempo, l'utilizzo di un Sistema che virtualizza le k risorse:

$$U_{Vi} = \sum_{j=1}^{k} U_{ji}$$

Considerando quindi la serie numerica data dalla somma di tutte le n colonne della matrice:

$$U_V = \left[\sum_{j=1}^{k} U_{j1} \;,\; \sum_{j=1}^{k} U_{j2} \;,\ldots \sum_{j=1}^{k} U_{jn} \right]$$

essa rappresenterà l'utilizzo della risorsa in grado di Virtualizzare le k risorse di partenza per tutti gli n intervalli considerati. L'utilizzo massimo che stiamo cercando, cioè quello che ci consentirà di determinare la capacità del Sistema Virtualizzato, sarà dato da:

$$\tilde{U}_V = \mathrm{Max}\left[\sum_{j=1}^{k} U_{j1} \;,\; \sum_{j=1}^{k} U_{j2} \;,\ldots \sum_{j=1}^{k} U_{jn} \right]$$

Ricordiamo ancora una volta che il massimo utilizzo così calcolato è sempre minore o uguale alla somma dei massimi utilizzi di ogni risorsa considerata.

Calcoliamo la capacità necessaria a consolidare una Server Farm costituita da N Serventi ciascuno con diversa Piattaforma Informatica. Un primo calcolo può essere basato sulla Frequenza di Clock: in questo caso raggruppiamo i serventi in L gruppi. In ciascun gruppo poniamo i serventi che hanno lo stesso utilizzo massimo indicato con con U_{bmax},

la stessa frequenza di clock che indichiamo con Ck_B e che svolgono lo stesso tipo di lavoro.

Contiamo il numero totale di processori (Cores) presenti in ogni gruppo; per ciascuno degli L gruppi si avrà:

$$[Nproc_Z]_h = \frac{[Ck_B * Nproc_B * U_{Bmax}]_h}{[WLF]_h * Ck_Z * U_Z}$$

Il numero di Cores del System z necessari e probabilmente sufficienti al consolidamento in base alla frequenza di clock sarà dato da:

$$Totproc_Z = \sum_{h=1}^{L} [Nproc_Z]_h = \sum_{h=1}^{L} \frac{[Ck_B * Nproc_B * U_{Bmax}]_h}{[WLF]_h * Ck_Z * U_Z}$$

Vediamo ora un esempio; vogliamo calcolare quanti processori System z di tipo IFL a frequenza di 1.8 Gigahertz usati al 100% sono necessari e probabilmente sufficienti a consolidare:

- 10 Sistemi x86 (B) con Clock di 1 Ghz, Utilizzo 60%, ciascuno con 4 cores e WLF=2
- 5 Sistemi x86 (C) con Clock di 3,5 Ghz, Utilizzo 60%, ciascuno con 2 cores e WLF=1

In questo esempio occorrono circa 19 processori (Cores) di tipo IFL.

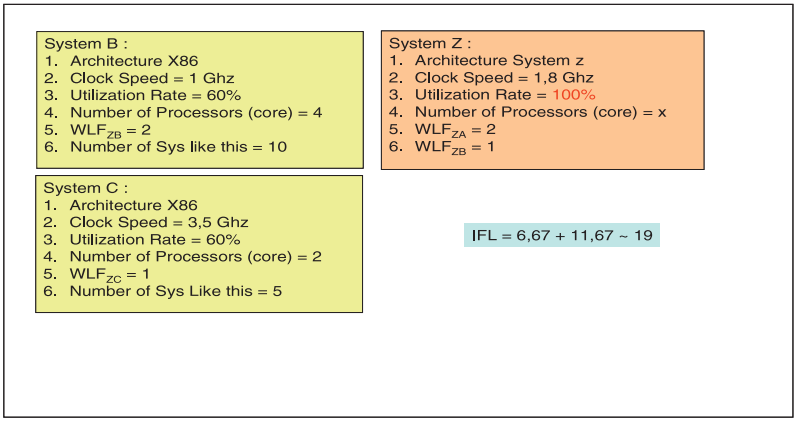


Figura 18 Esempio di calcolo di Sistemi equivalenti

Volendo eseguire un calcolo più accurato e disponendo del valore della equivalenza RPE→ MIPS (questo dato non è di pubblico dominio), si potrebbe operare in modo simile raggruppando tutti i Server con lo stesso WLF in gruppi e di questi sommando i valori di Rperf relativi (Rperf è proporzionale a RPE e tiene conto dell'equivalenza sopra detta) ottenendo k gruppi. In questo caso avremo:

$$MIPS_i = \frac{[rPerf_B * 100 * U_B]_i}{[WLF]_i * U_Z}$$

Questa formula vale per ogni intervallo elementare di tempo, come discusso precedentemente, quindi per ogni intervallo considerando tutti i gruppi (k), avremo:

$$TotMIPS_Z = \sum_{i=1}^{k} [MIPS]_i = \sum_{i=1}^{k} \frac{[rPerf_B * 100 * U_B]_i}{[WLF]_i * U_Z}$$

A questo punto applichiamo il metodo discusso in precedenza per il campionamento degli intervalli e calcoliamo la matrice degli utilizzi, ponendo in ciascuna riga i valori di ciascuno dei k gruppi per n intervalli di misura:

$$\tilde{U}_V = Max\left[\sum_{j=1}^{k} U_{j1} , \sum_{j=1}^{k} U_{j2} , \ldots \sum_{j=1}^{k} U_{jn} \right]$$

IL valore dei Mips totali ricercato sarà dato dal massimo delle somme così calcolate:

$$\widetilde{TotMIPS}_Z = \text{Max}\ [\ M_{j1}\ , M_{j2},\ \ldots..\ M_{jn},\ ]$$

in cui ciascun elemento sarà dato da una relazione del tipo:

$$M_j = \sum_{j=1}^{k} \frac{[rPerf_B * 100 * U_B]_j}{[WLF]_j * U_Z}$$

Il metodo illustrato è sufficientemente rigoroso ma richiede due importanti condizioni per essere accurato:

1. disporre di una misurazione significativa degli utilizzi dei Serventi di partenza, secondo le tecniche di campionamento sopra descritte;
2. disporre di valori accurati per le costanti WLF nell'intervallo di utilizzo misurato e di RPE ed Rperf.

Sconsigliamo fermamente di operare dimensionamenti a partire da dati di carico stimati e non misurati e senza avere una profonda conoscenza del tipo di lavoro. I valori di WLF possono essere misurati in modo sperimentale e poi utilizzati per casi simili.

4.0.8 Calcolo delle componenti di Costo [TCO]

Abbiamo visto come l'aspetto dei Costi compaia quasi sempre tra i Drivers della Platform Selection. L'analisi dei costi è quindi uno dei primi elementi da tenere in conto e richiede una formulazione quantitativa rigorosa che sia capace di fornire supporto alle decisioni. In altre parole lo scopo che ci proponiamo è quello di rispondere in maniera oggettiva alla domanda:
"Ho individuato una Piattaforma come possibile alternativa a quella attuale, ho individuato un driver di ottimizzazione (ad esempio il costo), quali sono le condizioni sotto le quali il processo di trasformazione individuato fornisce una convenienza in base al driver stabilito (costo inferiore) ed in quali tempi ottengo un ritorno dell'investimento (ROI) per la trasformazione ?"
Per rispondere efficacemente a questa domanda occorre disporre di strumenti quantitativi ed asettici che dovranno consentire di:

- valutare "a priori" l'efficacia del processo;
- fornire criteri di supporto alle decisioni;
- dare indicazioni sul ROI;
- verificare i risultati a posteriori.

Limitandoci al solo driver dei costi possiamo allora definire un metodo detto del Total Cost of Ownershp (TCO)[21] che definisce un costo:

TCO = TCOp + TCA

Il metodo ha l'obiettivo di confrontare due o più "soluzioni" dal punto di vista dei costi fornendo un modello assoluto ed indipendente da valutazioni soggettive. Il TCO viene calcolato e confrontato per tutte le soluzioni oggetto della analisi che poi verranno confrontate. Il TCO è dato dalla somma di due componenti diverse, una, detta TCOp, rappresenta i costi operativi ricorrenti e pertanto dipende dal tempo ed è variabile con esso:

$$\mathbf{TCOp} = \sum_{K=1}^{n} (\mathbf{TCOp})_K$$

dove $\mathbf{TCOp_k}$ rappresenta la componente dei costi ricorrenti per ogni periodo elementare, per il quale i costi sono ritenuti costanti (ad esempio un anno) e la sommatoria è estesa al periodi di n intervalli (ad esempio 3 anni). TCA invece rappresenta i costi di acquisizione una tantum, non dipende dal tempo ed è costante per tutto il periodo di valutazione. Poiché la grandezza TCO contiene una componente (TCOp) che è funzione del tempo, essa sarà funzione del tempo e quindi dovrà sempre essere riferita ad un intervallo di tempo prefissato. Tale intervallo di tempo è detto "Periodo di analisi".

Tipicamente in Italia si usano periodi di analisi di tre anni, cioè pari ai periodi di ammortamento dei beni informatici. Il TCO è un concetto intuitivo che può essere applicato a qualunque fatto della vita quotidiana: proviamo ad immaginare il TCO relativo al possesso di una automobile.

Nei costi ricorrenti (TCOp) calcoleremo ad esempio per ogni anno:

1. L'affitto del Box/garage
2. Il Costo dei Carburanti/Lubrificanti
3. Il Costo della manutenzione
4. Il Costo del lavaggio
5. La tassa di Proprietà
6. L'Assicurazione Obbligatoria

Nei costi di acquisizione (TCA), invece calcoleremo:

1. Costo di acquisto
2. Costo delle modifiche /Aggiunte
3. Costo dell'autoradio
4. Costo degli accessori

Potremmo quindi confrontare due automobili diverse calcolando nei due casi il TCO ad esempio per quattro anni (vita media dell'automobile). Per fare un calcolo accurato

[21] Il metodo fu introdotto alla fine degli anni novanta da alcuni consulenti indipendenti del mercato informatico e si basa sulla definizione di una quantità (espressa in valuta) denominata Total Cost of Ownership ed abbreviata con la sigla TCO.

dovremo aggiungere al TCA anche il valore residuo della automobile alla fine dei quattro anni (Net Present Value ovvero Ammortamento anticipato).

Nella pratica nessuno acquista o esclude l'acquisto di una automobile basandosi sul TCO; intervengono in questo caso "fattori esterni" di tipo emotivo che portano a intraprendere delle scelte indipendentemente dalla convenienza (moda, gradimento, diffusione sul mercato, ecc.). Questo tipo di fattori potrebbe anche influenzare la scelta o la selezione di una Piattaforma Informatica, ma ciò è proprio quello che stiamo cercando di evitare; se fattori non semplicemente quantificabili devono essere inseriti nel calcolo, essi devono essere trasformati in valori numerici o monetari. Il fatto che una Infrastruttura Informatica sia più stabile e resiliente di un'altra potrebbe essere trasformato in diminuzione dei "fermi non pianificati" e questo dovrebbe essere trasformato in diminuzione della perdita di denaro dovuta a questi fermi. In molti casi risulterà semplice effettuare una simile valutazione, in altri casi sarà praticamente impossibile.

Per esempio se la Piattaforma Informatica che stiamo analizzando ospita una Applicazione Informatica che svolge il ruolo di immettere e controllare le prenotazioni di una aerolinea tramite la rete, sarà facile determinare quale sia la perdita di passeggeri (denaro) dovuta al fatto che il servizio sia interrotto per un certo numero di ore: basterà conoscere quale sia in media il numero di posti/biglietti venduti ogni ora ed assumere che essi vadano tutti alla concorrenza nel caso in cui un utente non riesca a fare la prenotazione a causa di un fermo non pianificato. Molto più difficile o forse impossibile sarebbe valutare la perdita nel caso in cui la Piattaforma Informatica gestisca un servizio che non comporti esborso di denaro, ad esempio un servizio al cittadino da parte di un ente pubblico. Ci sono molti altri casi nei quali un fermo non pianificato non comporta alcun tipo di perdita in denaro.

Ciò premesso, possiamo limitarci, almeno in prima approssimazione, alle componenti di costo certe, lasciando ad una seconda analisi quelle più aleatorie o di difficile quantificazione.

Riferendoci alla Piattaforma Informatica o all'Infrastruttura Informatica sono elementi del TCOp:

- Costi di Personale (FTE – Full Time Equivalent)
- Costi di Energia Elettrica, Raffreddamento e Spazio (Environmental)
- Costi ricorrenti legati al Software (Manutenzione e Canoni)
- Costi ricorrenti legati all'Hardware (Manutenzione)
- Costi ricorrenti legati alla rete
- Altri costi Ricorrenti
- Valore delle interruzioni di Servizio periodiche
- .....

Sono invece elementi del TCA:

- Costi di acquisizione Hardware
- Costi di acquisizione Software
- Costi di acquisizione altri apparati
- Costo della migrazione dei Pacchetti Applicativi
- Consulenze

- Altre spese occasionali (Trasporto, Attrezzaggio, Montaggi)
- Costi di dismissione
- Net Present Value

Alcuni costi sono pesantemente influenzati dalla riduzione dei Sistemi Fisici, tra questi ricordiamo gli environmental, e alcuni costi legati alle licenze Software e alle manutenzioni. Il costo del personale (FTE) dipende in larga parte dal numero delle immagini di Sistema, cioè dei Serventi Logici.

Il metodo del TCO può essere usato nel confronto di due o più soluzioni informatiche (Piattaforme o Infrastrutture) secondo lo schema precentemente illustrato in Figura 12. In questo caso, se indichiamo con "old" lo stato attuale e con "new" quello ipotizzato dopo la trasformazione, utilizzando le definizioni fino ad ora enunciate avremo:

$$TCO_{OLD} = \sum_{i=1}^{K} (TCOp_{OLD})_i + TCA_{OLD}$$

e invece per la nuova Infrastruttura:

$$TCO_{NEW} = \sum_{i=1}^{K} (TCOp_{NEW})_i + TCA_{NEW}$$

E' bene osservare che il periodo di analisi, ovvero il valore di k, deve essere uguale nei due casi, se vogliamo operare un confronto. Una operazione di migrazione OLD → NEW risulterà conveniente in base al driver del TCO se accade che:

$$TCO_{OLD} > TCO_{NEW}$$

e quindi:

$$\sum_{i=1}^{K} (TCOp_{OLD})_i + TCA_{OLD} > \sum_{i=1}^{K} (TCOp_{NEW})_i + TCA_{NEW}$$

Se ci riferiamo ad una infrastruttura OLD esistente da un periodo maggiore di k, possiamo assumere che i costi di acquisizione a suo tempo sostenuti siano stati ammortizzati (ciò vale ad esempio per le valutazioni di casi di re-hosting).

Sotto queste condizioni possiamo porre:

$$TCA_{OLD} = 0$$

e quindi attraverso semplici passaggi risulterà:

$$TCA_{NEW} < \left[\sum_{i=1}^{K} (TCOp_{OLD})_i - \sum_{i=1}^{K} (TCOp_{NEW})_i \right]$$

La relazione sopraindicata esprime il concetto in base al quale una operazione di cambiamento di Piattaforma per una Infrastruttura già esistente è conveniente dal

punto di vista dei costi se il suo costo complessivo di acquisizione (espresso da TCA_{NEW}) risulta minore della differenza tra i costi operativi dell'Infrastruttura attuale e di quella nuova estesa al periodo di valutazione.

Il metodo illustrato fornisce risultati tanto più accurati quanto più accurata risulterà la determinazione delle quantità TCA e TCOp nei due casi. I concetti fino a qui esposti possono essere rappresentati in maniera grafica fornendo una rappresentazione visiva immediata del risultato.

La Figura 19 mostra un esempio di rappresentazione del TCO calcolato su un periodo di tre anni e per cinque diverse soluzioni alternative. Un esame visivo del grafico fornirà una indicazione su quale sia la piattaforma più conveniente dal punto di vista dei Costi.

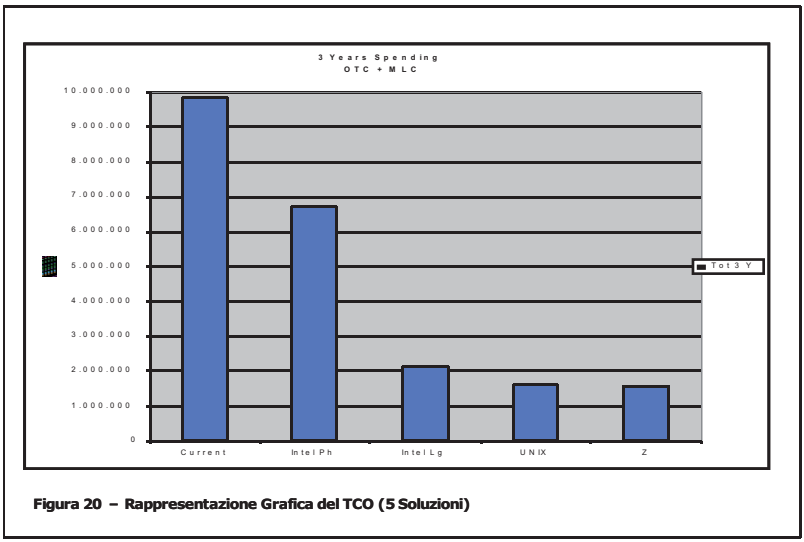


Figura 20 – Rappresentazione Grafica del TCO (5 Soluzioni)

Figura 19 Rappresentazione Grafica del TCO per 5 diverse Soluzioni

Si osservi che il TCO è composto da due componenti, una fissa nel tempo ed una variabile con esso: dal punto di vista economico/finanziario ciò si traduce dicendo che il TCO è la somma di una componente di Spese di Capitale (TCA) e di una di Costi Operativi (TCOp). Si definisce la prima con il termine inglese Capital Expenses (CAPEX) e la seconda con il termine inglese Operational Expenses (OPEX). Questa differenza potrebbe essere importante nella valutazione da parte di Enti o Società quotate in borsa e quindi andrebbe tenuta in considerazione.

Per completare la trattazione dell'argomento TCO occorre tener presente un ultimo aspetto, quello cioè del "Ritorno dell'Investimento" - ROI (Return of Investment). Il periodo di ROI o semplicemente il ROI è il periodo di tempo entro il quale

l'investimento operato per l'acquisizione di un bene o servizio viene ripagato dai risparmi che il bene o servizio produce con il suo uso. In altre parole, se la nuova infrastruttura presenta dei costi di esercizio TCOp inferiori a quelli attuali, si deve presumere che usando la nuova infrastruttura si possano ottenere dei risparmi. Però il passaggio dalla vecchia alla nuova infrastruttura comporta dei costi rappresentati dal TCA. Il periodo di ROI rappresenta l'intervallo di tempo a partire dal quale si comincerà a risparmiare, una volta recuperati investimenti o spese fatti per passare alla nuova infrastruttura.

In modo veloce ed intuitivo ricorriamo ad un metodo grafico, rappresentato dalla figura 20. In essa la linea rossa (Cum D) rappresenta la somma dei costi operativi correnti; tale linea è retta in quanto, al netto dei fattori inflattivi, i costi tendenzialmente sono costanti e quindi per ogni intervallo di tempo la somma cresce della stessa quantità. Poichè abbiamo ipotizzato che la nuova infrastruttura abbia dei costi operativi inferiori, essi saranno rappresentati da una retta con pendenza minore, nel nostro grafico la linea verde (Cum V). Le due rette tendono a divergere nel tempo: la distanza tra loro rappresenta il risparmio ottenuto. La linea verde infatti si trova sempre al di sotto della rossa. Poiché nella realtà il passaggio dall'attuale infrastruttura (linea rossa) alla nuova (linea verde) comporta spese/investimenti, dobbiamo sommare tali investimenti ai dati che hanno prodotto la linea verde; otterremo in questo modo la linea blu (Cum R) che rappresenta in termini economici le spese reali date dalla somma dei Costi Operativi più gli investimenti.

Per quanto detto risulta evidente che il periodo di ROI è dato dalla proiezione sull'asse dei tempi asse x del punto di incontro tra la linea rossa e quella blu. Poichè abbiamo dedotto che la distanza tra due linee rappresenta il risparmio tra due soluzioni, se consideriamo le linee rossa e blu, tale distanza è negativa (spesa) fino al punto di incontro; da lì in poi diventa (risparmio). Quindi l'ascissa del punto di incontro tra le due linee è il tempo di ROI.

Una simile tecnica può essere utilizzata anche per il confronto tra più di due soluzioni, in questo caso, come vedremo negli esempi, la linea verde non viene tracciata (essa in effetti non corrisponde ad una situazione reale), e si tracciano più "versioni" della linea blu per le tutte le alternative considerate rispetto alla situazione corrente. Concludiamo affermando che il metodo illustrato fornisce, in maniera quantitativa rispetto al driver del costo, indicazioni su come scegliere la Piattaforma o l'infrastruttura migliore.

Il metodo, la cui applicazione sarà illustrata da alcuni esempi, potrà essere utilizzato:

1. per confrontare due o più piattaforme informatiche dal punto di vista dei costi
2. per valutare processi di re-hosting determinandone o meno la convenienza
3. per giustificare l'uso di Linux su Sistemi Centrali
4. per giustificare processi di Server Consolidation[22]
5. in generale in quasiasi processo di ITRO che abbia la riduzione dei costi come driver.

[22] Vedi Paragrafo 4.0.13

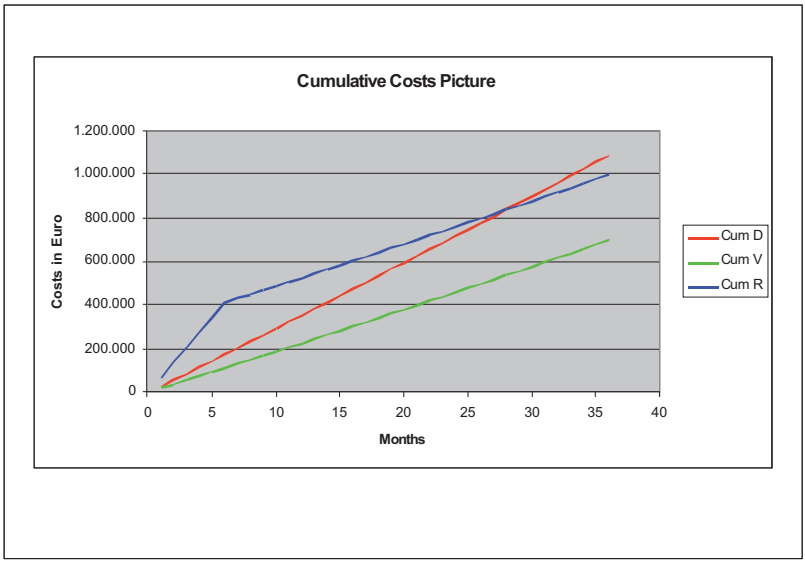


Figura 20 Rappresentazione Grafica del ROI per due Soluzioni

4.0.9 Un Esempio: Scelta della Piattaforma [23]

Vogliamo attivare presso la nostra Organizzazione una applicazione di Business Analytics Optimization (BAO) da realizzare ex-novo con l'uso di un prodotto Software disponibile sul mercato su quattro piattaforme differenti (che indicheremo con PLATF1, PLATF2, PLATF3, PLATF4).

Premesso che: il costo della realizzazione della Applicazione sia uguale sulle quattro piattaforme considerate, le funzioni siano garantite in tutti e quattro i casi e il Personale necessario e sufficiente a gestire l'ambiente operativo sia lo stesso per tutte e quattro le piattaforme, vogliamo determinare quale sia la Piattaforma Informatica più conveniente ad ospitare la nostra applicazione BAO.

Procediamo quindi alla determinazione dei Sistemi necessari ad ospitare l'applicazione nei quattro casi. Il fornitore del Software ci ha indicato con il metodo del lavoro equivalente i Sistemi minimi in grado di soddisfare le nostre esigenze prestazionali; siamo quindi in grado di determinare con precisione quali siano i quattro Sistemi da considerare.

[23] I Dati di Costo sono fittizi e pertanto non rappresentano valori reali. L'esempio serve solo a descrivere il metodo e non è indicativo del risultato.

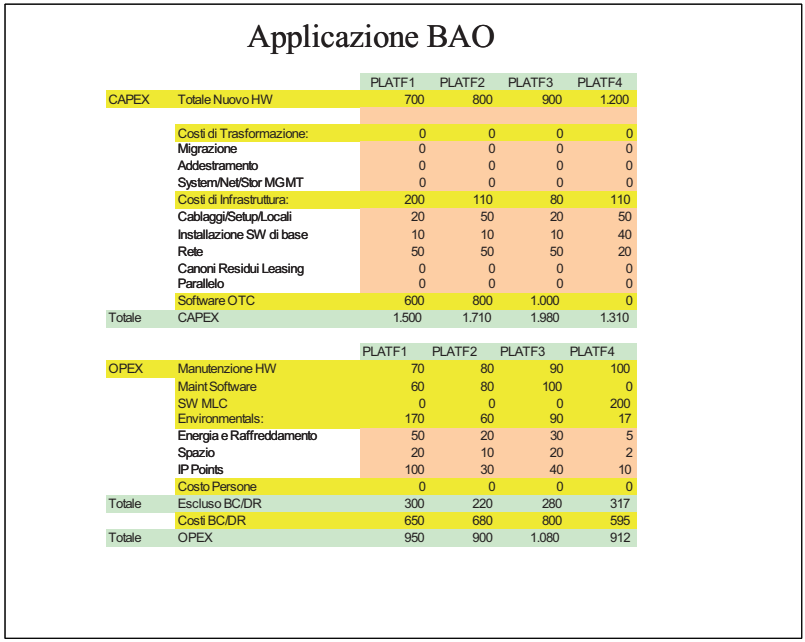
Per ciascuno di essi determiniamo:

1. Costo di Acquisizione dell'Hardware
2. Costo di Acquisizione del Software
3. Canoni di Manutenzione dell'Hardware
4. Canoni di Manutenzione del Software
5. Software a Canone Mensile
6. Costi di Energia Elettrica e Raffreddamento
7. Costi dello Spazio e dei Punti Rete
8. Costi di Attrezzaggio ed Impianti.

Alcuni costi sono costanti, ad esempio il costo dell'applicazione, dell'addestramento e del Personale di Gestione: in questo caso ai fini della nostra analisi è opportuno trascurarli in quanto invarianti e quindi nella successiva tabella saranno messi uguali a zero. Teniamo conto invece dei costi per la continuità operativa (BC – Business Continuity) e per il ricovero di eventuali disastri (DR – Disaster Recovery), che valutiamo in percentuale dei Costi Operativi e dei Costi di Acquisizione, ipotizzando di stipulare allo scopo un contratto di Servizio con un Terzo (per questa ragione saranno posti sotto la voce dei Costi Operativi). Tale valutazione viene fatta seconda la seguente formula:

Costo_DR(Anno) = CAPEX/3 + OPEX/2

Con queste ipotesi costruiamo la tabella indicata in Figura 21, in essa solo le righe in giallo vengono sommate per fornire i totali (in verde), le righe in arancio rappresentano invece il dettaglio delle righe in giallo. Osserviamo che l'andamento dei Costi nei quattro casi si presenta molto variabile; ad esempio la PLATF4 presenta costi di acquisizione dell'hardware nettamente maggiori. Come già detto, tutti i costi che sono uguali nei quattro casi sono stati posti a zero (cioè trascurati).



Applicazione BAO

		PLATF1	PLATF2	PLATF3	PLATF4
CAPEX	Totale Nuovo HW	700	800	900	1.200
	Costi di Trasformazione:	0	0	0	0
	Migrazione	0	0	0	0
	Addestramento	0	0	0	0
	System/Net/Stor MGMT	0	0	0	0
	Costi di Infrastruttura:	200	110	80	110
	Cablaggi/Setup/Locali	20	50	20	50
	Installazione SW di base	10	10	10	40
	Rete	50	50	50	20
	Canoni Residui Leasing Parallelo	0	0	0	0
		0	0	0	0
	Software OTC	600	800	1.000	0
Totale	CAPEX	1.500	1.710	1.980	1.310

		PLATF1	PLATF2	PLATF3	PLATF4
OPEX	Manutenzione HW	70	80	90	100
	Maint Software	60	80	100	0
	SW MLC	0	0	0	200
	Environmentals:	170	60	90	17
	Energia e Raffreddamento	50	20	30	5
	Spazio	20	10	20	2
	IP Points	100	30	40	10
	Costo Persone	0	0	0	0
Totale	Escluso BC/DR	300	220	280	317
	Costi BC/DR	650	680	800	595
Totale	OPEX	950	900	1.080	912

Figura 21 Tabella dei Costi per l'applicazione BAO

La figura 21 consente di determinare il grafico dei costi di acquisizione TCA (figura 22) e dei costi Operativi annuali (TCOp)[24] – figura 23. Considerato infine un periodo di analisi di Tre Anni, possiamo ottenere una figura del TCO (figura 24).

[24] I costi operativi annuali sono ipotizzati costanti nel tempo. Tale approssimazione potrebbe risultare non accurata a causa della presenza di Garanzia sui prodotti Hardware o Software, in questo caso sarebbe necessario differenziare i costi di ogni anno.

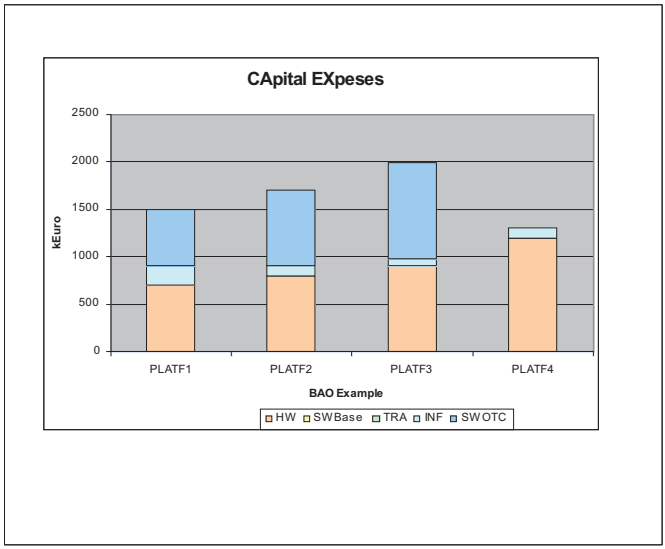


Figura 22 Costi di Acquisizione Applicazione BAO

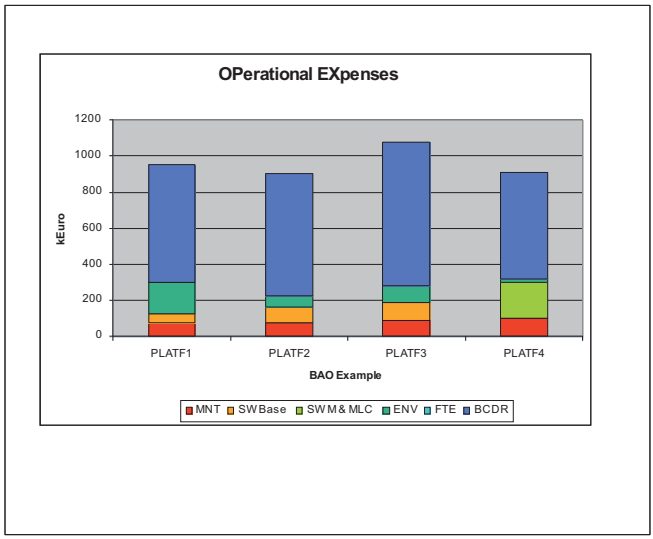


Figura 23 Costi Operativi Applicazione BAO

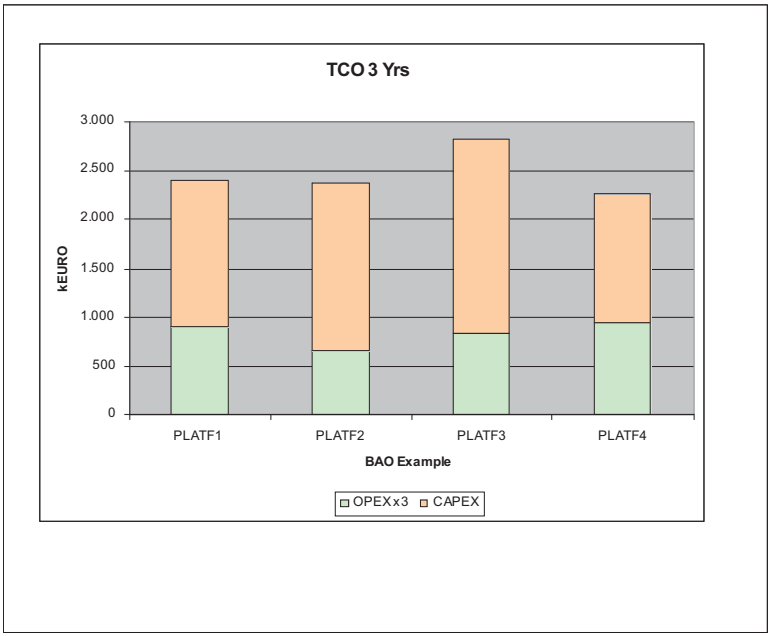


Figura 24 Total Cost of Ownership TCO (3 Anni) Applicazione BAO

Il metodo ci ha fornito informazioni sulla Scelta della Piattaforma Informatica considerando il driver dei costi, non la "prima impressione": la PLATF4 risulta essere la più conveniente su un periodo di tre anni, nonostante presenti costi dell'hardware maggiori e costi di esercizio (senza BC/DR) sensibilmente maggiori delle altre.

E' ovvio che, modificando il periodo di analisi, il risultato può cambiare anche **r**adicalmente: a tal proposito osserviamo la figura 25. Da essa si evince una indicazione sull'andamento del TCO negli anni e sulla convenienza di ciascuna Piattaforma Informatica nel tempo[25]: essendo la linea della PLATF3 (in giallo) sempre al di sopra delle altre, tale Piattaforma risulterà sempre la più costosa. La PLATF4 (in azzurro) invece, inizialmente più conveniente (nel periodo esaminato di Tre anni), perderebbe la sua convenienza in favore della PLATF2 (in rosso) su un periodo di Sei anni, ma non raggiungerebbe prima di Dieci anni la PLATF1 (in blu), che invece risulta più conveniente della PLATF2 su un periodo di Tre anni.

[25] Osserviamo che il calcolo mostrato assume che i costi siano costanti negli anni, tale ipotesi è in generale non corretta, soprattutto su lunghi periodi di tempo, ma rappresenta qui una buona approssimazione didattica: basterà nella reltà introdurre opportune correzioni che tengano conto degli effetti inflattivi e della variabilità dei costi di manutenzione negli anni.

La scelta di una piattaforma informatica non deriva solo da considerazioni sui costi: il modello che abbiamo utilizzato consente però di avere un quadro chiaro e di evitare scelte avventate o basate su "sensazioni" piuttosto che su dati di fatto. Il periodo di analisi deve essere scelto in base a parametri economico-finanziari, come il periodo di ammortamento dei beni informatici o a parametri organizzativi, come ad esempio la vita media stimata della Applicazione Informatica considerata.

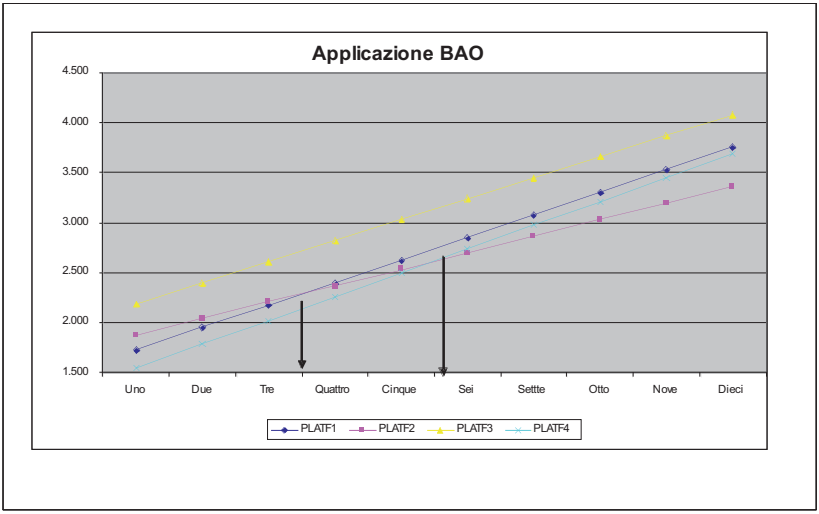


Figura 25 Spesa Annuale Cumulata - Applicazione BAO

4.0.10 Un Esempio: Cambio di piattaforma[26]

Un utente di una complessa Applicazione industriale ha la sensazione che la sua Infrastruttura Informatica, basata principalmente su un Sistema Centrale, sia eccessivamente "costosa" e pensa quindi di cambiarla operando un re-hosting delle sue applicazioni. Poiché si tratta di una struttura complessa, non tutto il Software che costituisce le Applicazioni risulta portabile. L'utente opera pertanto una prima analisi tecnica del suo Software per verificare la sua portabilità ed ottiene così il valore della Spesa necessaria al processo di re-hosting.

[26] I Dati di Costo sono fittizi e pertanto non rappresentano valori reali. L'esempio serve solo a descrivere il metodo e non è indicativo del risultato.

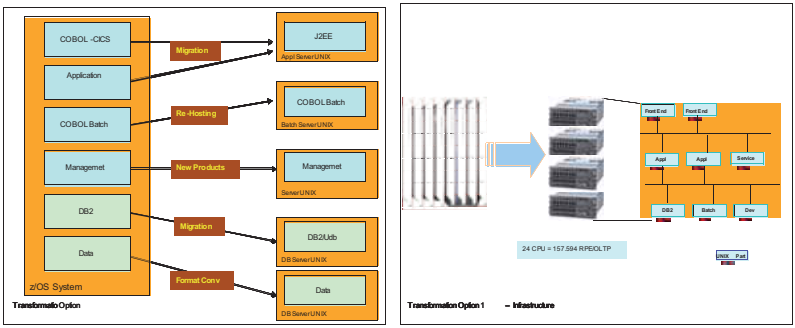
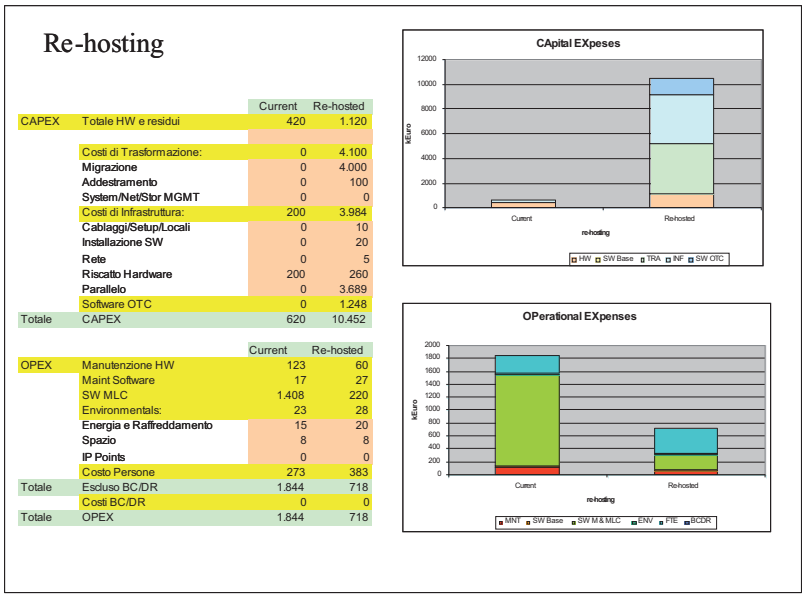


Figura 26 Ipotesi di Re-hosting

La piattaforma target viene individuata in IBM AIX + RISC, il Sistema Target è stato calcolato col metodo del Sistema Equivalente.

La figura 26 mostra il cammino di porting da effettuare. Dopo aver valutato i costi della operazione secondo la metodologia già nota, osserviamo che la soluzione re-hosted (figura 27) presenta un TCOp sensibilmente più basso della soluzione corrente e questa era la ragione che aveva spinto l'utente a valutare l'operazione di re-hosting.

Tuttavia l'operazione di re-hosting (figura 28) presenta un TCO a Tre anni sensibilmente maggiore dei costi attuali: ci poniamo quindi il problema di determinare dopo quanto tempo la soluzione re-hosted possa dare una convenienza.



		Current	Re-hosted
CAPEX	Totale HW e residui	420	1.120
	Costi di Trasformazione:	0	4.100
	Migrazione	0	4.000
	Addestramento	0	100
	System/Net/Stor MGMT	0	0
	Costi di Infrastruttura:	200	3.984
	Cablaggi/Setup/Locali	0	10
	Installazione SW	0	20
	Rete	0	5
	Riscatto Hardware	200	260
	Parallelo	0	3.689
	Software OTC	0	1.248
Totale	CAPEX	620	10.452

		Current	Re-hosted
OPEX	Manutenzione HW	123	60
	Maint Software	17	27
	SW MLC	1.408	220
	Environmentals:	23	28
	Energia e Raffreddamento	15	20
	Spazio	8	8
	IP Points	0	0
	Costo Persone	273	383
Totale	Escluso BC/DR	1.844	718
	Costi BC/DR	0	0
Totale	OPEX	1.844	718

Figura 27 Confronto Costi Re-Hosting

La Curva del ROI (figura 29) mostra che l'operazione non risulta genericamente conveniente in quanto il ROI viene pesantemente spostato in avanti (circa 10 anni) a causa dei grandi investimenti che occorre fare: tali investimenti sono dovuti all'alto costo della migrazione e ad un periodo di parallelo (2 anni) nel quale occorrerà mantenere attive entrambe le soluzioni.

Quindi, pur essendo di fronte ad una soluzione che presenta costi di esercizio TCOp più basso degli attuali in processo di re-hosting, nel caso trattato questa soluzione non risulta conveniente.

Sarebbe forse opportuno in questo caso adottare una soluzione di mantenimento della Piattaforma Informatica corrente, modificando le applicazioni in modo da diminuire i costi di esercizio (ad esempio riducendo le licenze Software), ottenendo un ROI in tempi più brevi.

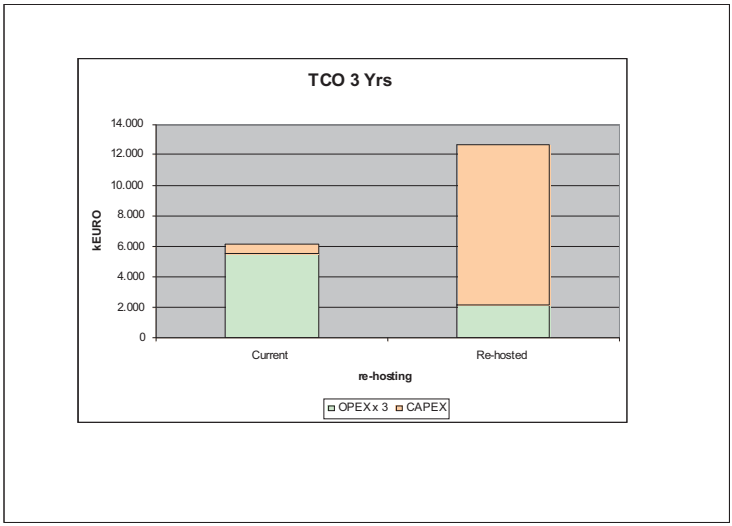


Figura 28 TCO Re-Hosting

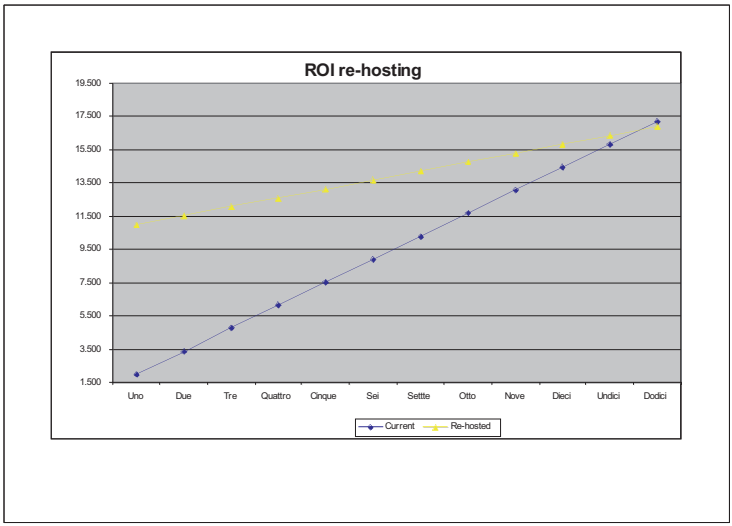


Figura 29 ROI Re-Hosting

4.0.11 Ottimizzazione della Infrastruttura Informatica

Abbiamo già definito ***Infrastruttura Informatica*** come ***l'ambiente di esecuzione di una o più Applicazioni Informatiche*** – vedi figura 7**.**

Per come definita, nell'Infrastruttura Informatica potranno essere presenti Piattaforme diverse ed inoltre una Infrastruttura Informatica potrà eseguire una o più Applicazioni – vedi figura 13.

L'insieme degli interventi atti a rendere l'infrastruttura Informatica più semplice, più stabile, più efficiente e più gestibile prende il nome di **Ottimizzazione della Infrastruttura Informatica -** ITRO (Information Technology Resources Optimization).

Prima di affrontare il processo di ITRO sarà opportuno analizzare quali sono le cause riconosciute della complessità dell'Infrastruttura e quali i problemi più importanti e di maggiore impatto su di essa.

A partire dalla metà degli anni '80, il Modello Applicativo denominato "Client Server" ha creato una proliferazione abbastanza incontrollata di Serventi (Elaboratori), che venivano installati in grande numero ed in locazioni differenti.

Lo stesso modello applicativo ha spesso introdotto il paradigma: "Una Funzione = Un Servente" che poi col passare degli anni è diventato "Una Funzione = Più Serventi", da ciò deriva che il numero di serventi installati è diventato molto grande, finendo col costituire esso stesso un problema infrastrutturale.

Sempre a causa del Modello Applicativo ed in relazione ad una scarsa pianificazione della Infrastruttura, questa ha finito con l'includere Piattaforme diverse e molteplici che risiedono in locazioni geografiche distanti tra loro: questo di per sé non può considerarsi un "errore di progettazione", ma il risultato inevitabile di carenze tecnologiche e debolezze dei Sistemi Operativi.

Anche se oggi queste carenze sono state completamente superate e la tecnologia dei Sistemi Operativi è notevolmente progredita, il risultato di un paio di decenni di proliferazione incontrollata è ancora sotto i nostri occhi e continua a produrre gli stessi disagi.

Pur avendo preso coscienza del problema e pur disponendo oggi delle tecnologie adeguate, in assenza di un intervento specifico si rischia di peggiorare ulteriormente la situazione.

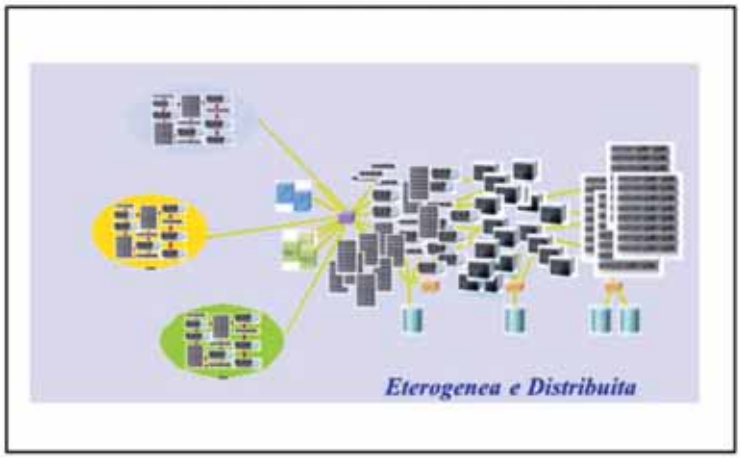


Figura 30 L'infrastruttura è oggi elemento di complessità

Possiamo quindi affermare che ancora oggi, anche a causa del permanere di un grande numero di Serventi Eterogenei e distribuiti, l'infrastruttura rimane un elemento di grande complessità del quale ci dobbiamo occupare (figura 30).

Un processo di ITRO in altri casi può avvenire in modo preventivo, cioè all'atto della definizione di una nuova infrastruttura, ma potrebbe anche essere effettuato "a posteriori" su una infrastruttura già esistente e produttiva: possiamo affermare anzi che quest'ultimo caso è quello più frequente nella pratica. La Proliferazione dei Serventi non è l'unico problema riscontrabile nell'Infrastrura Informatica. Lo stesso modello applicativo che ha creato la proliferazione - ogni servente svolge sempre una sola funzione - ha generato anche uno scarso utilizzo dei Sistemi Serventi: i Sistemi col tempo sono diventati sempre più potenti e di conseguenza, dovendo e potendo svolgere una sola funzione, sono diventati sempre meno utilizzati. Anche in questo caso non si tratta di un "errore" di progetto, ma di una semplice conseguenza del disegno architetturale delle applicazioni e dei limiti dei Sistemi Operativi, i quali spesso non consentono lo svolgimento di funzioni diverse sulla stessa Istanza di Sistema Operativo. Un tema collaterale che ne deriva è quello dei Costi Gestionali (Cost of Management), che crescono col crescere del numero dei Sistemi e componenti installati e in esercizio.

Un discorso a parte andrebbe fatto per i problemi legati alla **Qualità del Servizio** (Quality of Service), alla **Resilienza** (Resiliency) e alla **Ricoverabilità in caso di Disastro** (Disaster Recovery).

Sembra intuitivo il fatto che la possibilità di interruzione accidentale del servizio è funzione del numero di componenti discrete della Infrastruttura e quindi la riduzione del numero di tali componenti possa diminuire le interruzioni: poichè la **Qualità del Servizio** è inversamente proporzionale al numero di fermi accidentali, ne viene che una Infrastruttura che presenta un minor numero di componenti discreti presenterà anche una migliore Qualità del Servizio.

La **Resilienza** è la capacità dell'Infrastruttura di "reagire" dinamicamente alle variazioni del carico applicativo, vale a dire l'elasticità che essa presenta rispetto ad una sollecitazione di intensità variabile: si può intuire come un'infrastruttura con un numero minore di Sistemi reagisca meglio a sollecitazioni esterne, a causa della possibilità dei serventi di riutilizzare le capacità elaborative lasciate libere dalle altre applicazioni.

Negli anni più recenti stanno emergendo esigenze di tipo ambientale legate al contenimento dei consumi energetici, delle necessità di raffreddamento e alla carenza di spazio: avere un grande numero di Sistemi in generale poco utilizzati implica la necessità di grandi potenze elettriche, grandi dissipazioni di calore e l'occupazione di grandi spazi; questo può diventare un elemento di grande criticità soprattutto quando si ha carenza nella disponibilità di Energia Elettrica o di Spazio. Uno dei temi legati all'Infrastruttura è quindi anche quello della **riduzione dell'impatto ambientale**: molti apparati, soprattutto quelli più "datati", presentano scarsa efficienza energetica e necessitano di molta energia per essere raffreddati: indipendentemente dall'aspetto legato strettamente ai costi dell'energia ciò può avere anche una evidente ricaduta sull'inquinamento e sulla produzione di Anidride Carbonica.

Inoltre alcuni paesi del mondo occidentale cominciano ad avere, soprattutto nelle aree metropolitane densamente popolate, problemi di scarsità di approvvigionamento di energia elettrica in alcune ore del giorno: ciò impone la necessità di limitare i consumi e quindi di migliorare l'efficienza energetica a parità di potenza di calcolo. Tutto l'insieme degli interventi volti a migliorare l'impatto ambientale della Infrastruttura Informatica prende il nome di "Green Computing" e i Centri di Elaborazione Dati che contengono Infrastrutture a basso impatto ambientale vengono detti "Gren Data Center".

4.0.12 Fattori che influenzano il processo di ITRO

Il processo di ITRO in generale è lungo e può essere molto complesso; esso può avvenire intervenendo su fattori ed elementi differenti, ciascun elemento che interviene nel processo viene comunemente detto driver; il processo può compiersi in varie fasi, e può essere facilitato dalla tecnologia.

I driver più comunemente legati al processo di ITRO sono:

- Efficienza: si può intervenire sull'Infrastruttura cercando di migliorarne l'efficienza, indipendentemente dagli altri fattori.
- Gestibilità: si può intervenire sull'Infrastruttura solo per renderla più gestibile.

- Costi: si può operare sull'Infrastruttura al fine di diminuirne i costi operativi e di acquisizione; questo in effetti è il driver che più frequentemente spinge i processi di Ottimizzazione.
- Resilienza o Continuità di Servizio: si può operare sulla Infrastruttura solo per migliorarne le capacità di essere ricoverata in caso di disastro o la sua flessibilità operativa.
- Consumi Energetici: prende sempre più consistenza la valutazione degli impatti ambientali come driver per l'ottimizzazione.

Il tema dei costi è il driver che innesca il maggior numero di processi di ITRO. Una volta determinato il o i driver che innescano il processo, occorrerà determinare quali siano le Azioni da intraprendere o da valutare ai fini del processo stesso. Ad esempio azioni possibili potranno essere:

1. Diminuire il Numero di Serventi.
2. Consolidare gli apparati a disco.
3. Diminuire il numero di Istanze di Sistema Operativo presenti nella infrastruttura.
4. Virtualizzare i Sistemi.
5. Virtualizzare le Reti trasmissive.
6. Usare Sistemi Operativi meno costosi a parità di funzioni.

Le azioni sopra indicate, che non sono le uniche possibili, possono essere facilitate dalla tecnologia; citiamo ad esempio:

- La Virtualizzazione, che consente di ridurre il numero di Serventi Fisici e spesso anche il numero di quelli logici con poco sforzo operativo (**Server Consolidation**).
- L'uso del Sistema Operativo **Linux**, generalmente poco costoso e facilmente manutenibile.
- L'uso di **Grandi Sistemi Centrali** al posto di molti serventi distribuiti; tale uso, abbinato alla virtualizzazione, può consentire grossi recuperi di efficienza complessiva.

4.0.13 Il Consolidamento dei Serventi

Si definisce "Server Consolidation" il processo volto a diminuire (consolidare) il numero di Serventi Fisici o Logici componenti l'infrastruttura Informatica. Tale processo potrebbe essere complesso o di difficile realizzazione a causa di vincoli applicativi. Per tale ragione si indica con "Server Consolidation" il processo che tende a diminuire il numero di Serventi Fisici o Logici di una Infrastruttura Informatica con interventi di modifica piccoli, al limite nulli, sulle Applicazioni Informatiche che su di essa insistono. Tale operazione può essere effettuata sia sui Serventi Fisici che sui Serventi Logici; essa contribuisce al processo di Riduzione dei Costi Operativi e quindi è parte del processo di ITRO, di cui molto spesso rappresenta la parte più rilevante.

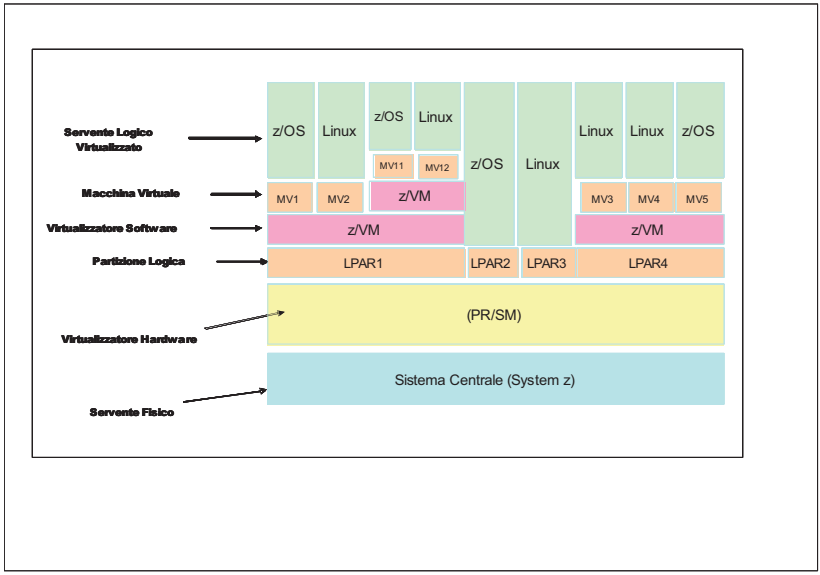


Figura 31 Riepilogo Virtualizzazione

Il processo di Server Consolidation è agevolato dalle capacità di virtualizzazione dei Serventi ed ha senso e si presenta più efficace in quanto spesso i Serventi di partenza (cioè quelli da consolidare) sono molti e singolarmente poco utilizzati. Diversamente, nel caso in cui i Serventi di partenza fossero pochi e molto utilizzati, esso risulterebbe un processo poco rilevante al fine della diminuzione della potenza complessivamente necessaria. Queste affermazioni sono dimostrate da quanto abbiamo affermato in precedenza a proposito dei vantaggi della virtualizzazione in termini di potenza complessiva richiesta per un insieme di Serventi che eseguono la stessa Applicazione Informatica.

La Virtualizzazione è fondamentale per realizzare la Server Consolidation e più in generale per i processi di ITRO, in quanto consente di diminuire il numero di Serventi Fisici e Logici e di utilizzarli al meglio. La figura 32 riepiloga i concetti di base legati alla virtualizzazione; ecco alcune definizioni di base in termini di Virtualizzazione:

- **Servente Fisico** o Sistema Fisico è un calcolatore elettronico fisicamente indipendente ed isolato, ovvero dotato di alimentazione elettrica indipendente, costituito da componenti elettricamente isolate ed indipendenti da altri Sistemi.
- **Servente Logico** o Immagine di Sistema è una istanza di Sistema Operativo indipendente, cioè attivabile o disattivabile in modo non correlato alla sua collocazione su un Servente Fisico.

- Un **Servente** è **Virtualizzato** se esso è attivato su un Servente Fisico insieme ad altri Serventi Logici: in tal caso il Servente Fisico disporrà di una apposita componente Hardware o Software o di entrambe, detta "virtualizzatore", in grado di condividere il Servente Fisico tra più Serventi Logici (Istanze di Sistema Operativo).
- Diremo "**Macchina Virtuale**" o **Partizione Logica** la porzione o l'insieme di risorse di un Servente Fisico dinamicamente assegnate da un dispositivo virtualizzatore ad un Servente Logico Virtualizzato.

Quindi un Servente Virtualizzato è solo e sempre un Servente Logico e inoltre su un Servente Fisico possono essere attivati più Serventi Logici mediante uno o più virtualizzatori. Definiamo ora il processo di Server Consolidation di cui indichiamo alcuni fasi (vedi figura 32):

- **Accentramento** (o Centralizzazione): caso in cui svariati serventi fisici posti in diverse locazioni geografiche vengono "accentrati" in una sola di esse. L'accentramento è una pratica sempre più frequente e si contrappone a quella della "distribuzione geografica" particolarmente in uso negli anni '80. Essa è resa possibile dalle accresciute capacità di banda delle reti trasmissive e dai diminuiti costi delle linee di comunicazione, nonchè dalle nuove tecnologie trasmissive che eliminano i problemi legati alla difficoltà di comunicazione tra i Client distribuiti e i Server centralizzati. L'accentramento presenta una convenienza "a priori", indipendentemente dai maggiori costi indotti dalle linee di comunicazione, in quanto concentra in una o due locazioni geografiche tutti i Serventi e si ottiene il vantaggio di non dovere spostare persone (o avere persone in diverse località) per gestire i Sistemi; inoltre si possono realizzare importanti sinergie in termini di potenza e di Continuità di Servizio. L'accentramento è una pratica ormai diffusa e consolidata.

- **Consolidamento Fisico** (Physical Consolidation) è il processo per cui diverse immagini di Sistema Operativo, prima ospitate su Serventi Fisici separati, vengono consolidate su un unico Servente Fisico. Tale processo è tanto più efficace quanto più le capacità di virtualizzazione e di condivisione delle risorse del Sistema Fisico ricevente sono sofisticate. Ad esempio, se il Sistema Fisico ricevente è in grado di virtualizzare con granularità molto "fine" sarà possibile consolidare un grande numero di Serventi Logici in pochi Serventi Fisici altamente utilizzati. Viceversa, se il Sistema Fisico ricevente presenterà un basso livello di condivisione delle risorse, saranno ottenuti risultati modesti in termini di risparmio complessivo di potenza elaborativa. Il Consolidamento Fisico è la pratica maggiormente diffusa nella realtà produttiva ed è spesso il massimo livello a cui si può giungere senza essere costretti a operare interventi profondi sulle applicazioni informatiche.

- **Consolidamento Logico** (Logical Consolidation) rappresenta il più efficace processo di consolidamento, in quanto si pone l'obiettivo di ridurre le immagini di Sistema installate (Sistemi Logici). Esso consiste nel mettere il maggior numero di funzioni applicative nello stesso Sistema Logico. Il consolidamento logico è una pratica complessa in quanto ai Sistemi Operativi richiede particolari caratteristiche, come la capacità di ospitare diverse funzioni e carichi eterogenei. Questa possibilità non è presente in

tutti i Sistemi Operativi. Il Consolidamento Logico rappresenta un importante elemento di ottimizzazione poichè riduce notevolmente il numero di immagini di Sistema da gestire.

La Figura 33 mostra la differenza tra i processi di consolidamento possibili. Dal punto di vista dell'Infrastruttura, la sola fondamentale differenza tra i processi di consolidamento Fisico e Logico è rappresentata dal fatto che nel primo caso ogni funzione di servente continuerà a risiedere sulla stessa immagine di Sistema Operativo dalla quale proveniva, nel secondo caso invece più funzioni di Servente Logico verranno accorpate sotto il controllo dello stesso Sistema Operativo e ciò potrà essere possibile solo se il Sistema Operativo ricevente è in grado di ospitare contemporaneamente diversi carichi eterogenei.

Nei processi di Consolidamento Logico è essenziale che i sistemi operativi possano ospitare sotto la stessa istanza applicazioni diverse contemporaneamente; non tutti i Sistemi Operativi possono farlo; al contrario la Virtualizzazione consente di operare consolidamenti Fisici sullo stesso Servente Fisico di diverse Istanze di Sistema Operativo (Serventi Logici) senza intervenire su alcuna caratteristica dei Sistemi Operativi che restano comunque inalterate; tale processo è di gran lunga più semplice.

Concludendo, il Consolidamento dei Serventi (Server Consolidation) può essere il primo e decisivo passo per un processo di ITRO; esso può avvenire per gradi e con livelli di intervento diversi.

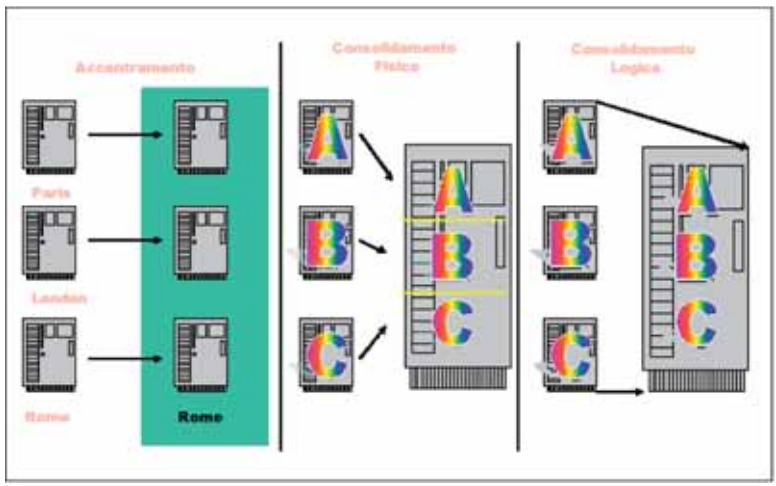


Figura 32 Tipi di Server Consolidation

Il Consolidamento dei Serventi può essere considerato ormai una necessità a causa della complessità raggiunta dalle Infrastrutture Informatiche; anche esso pone la necessità di disporre di elementi quantitativi in grado di valutare il processo e di stabilire se e quando esso sia conveniente in base ai driver di ottimizzazione stabiliti.

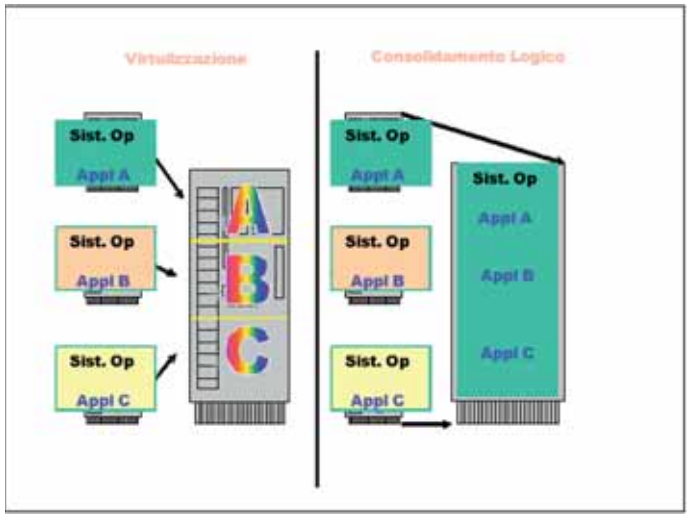


Figura 33 Virtualizzazione e Consolidamento Logico

4.0.14 Un Esempio: Valutazione di un processo di Server Consolidation [27]

L'ultimo esempio che esponiamo riguarda una Applicazione di sportello Bancario che oggi è installata su circa 1500 serventi di Agenzia, ciascuno localizzato presso le diverse Agenzie della Banca.

L'applicazione risale agli anni ottanta, quando la scarsa efficienza delle reti trasmissive imponeva la presenza dei Dati sui Sistemi Locali (Server di Agenzia) posti presso ognuna delle Agenzie della Banca in locazioni geografiche diverse e distanti; tale soluzione consentiva l'operatività della Agenzia anche in assenza di connessione col Sistema Centrale: oggi questa necessità è praticamente venuta meno.

[27] I Dati di Costo sono fittizi e pertanto non rappresentano valori reali. L'esempio serve solo a descrivere il metodo e non è indicativo del risultato.

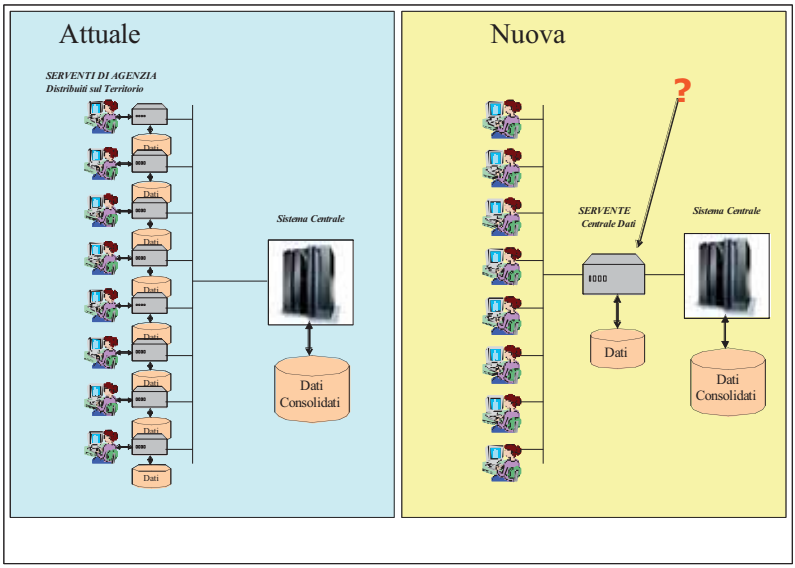
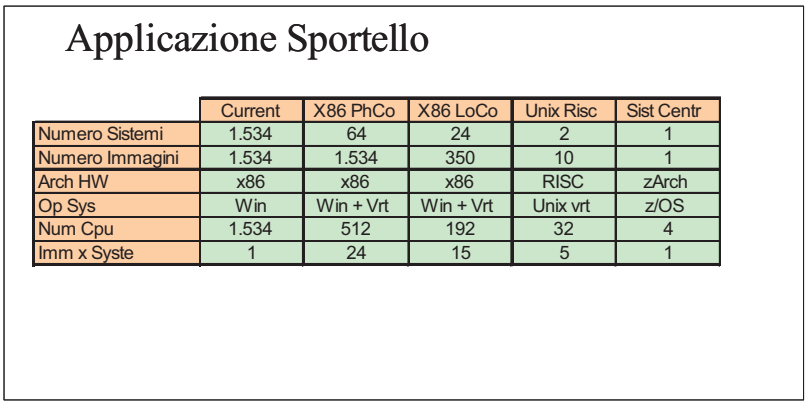


Figura 34 Infrastruttura Applicazione Sportello

Sembra quindi ovvio pensare ad un processo di ottimizzazione che elimini tutti i Server di Agenzia che contengono dati, accentrandoli su uno o più Sistemi posti nella locazione principale sede della Banca: tale processo elimina la necessità di gestire migliaia di macchine distribuite sul territorio e quindi appare di sicura convenienza economica, oltre che di sicuro miglioramento operativo. La figura 34 mostra lo scenario attuale e quello migliorativo che si vuole realizzare. Il problema che vogliamo risolvere è quello di determinare la migliore Soluzione dal punto di vista dei Costi che realizzi il nostro obiettivo. Poiché l'applicazione in questione è realizzata secondo il modello Client-Server e l'accesso al database avviene con chiamate di tipo 'SQL Standard', è realistico ritenere che non vi siano legami di nessun tipo tra l'Applicazione e gli Operating Environment: siamo quindi autorizzati ad avviare un processo di Selezione completamente libero da ogni vincolo applicativo. Per comodità individuiamo quattro possibili alternative[28]:

1. Accentramento dei circa 1500 Serventi su Sistemi della Stessa Architettura Hardware, mantenendo la stessa Piattaforma ma operando una Virtualizzazione a parità di numero di immagini di Sistema.
2. Consolidamento Logico con riduzioni di Immagini di Sistema su Grandi Sistemi Virtualizzati mantenendo la stessa Piattaforma Informatica.
3. Consolidamento Applicativo su grandi Sistemi UNIX-Risc Virtualizzati con riduzione di Immagini di Sistema.
4. Consolidamento Applicativo su un Unico Sistema Centrale su un database risiedente su uno dei Sistemi Centrali già in uso della Banca.

[28] Le quattro alternative individuate rappresentano le scelte più logiche e plausibili in relazione alla infrastruttura attualmente presente.

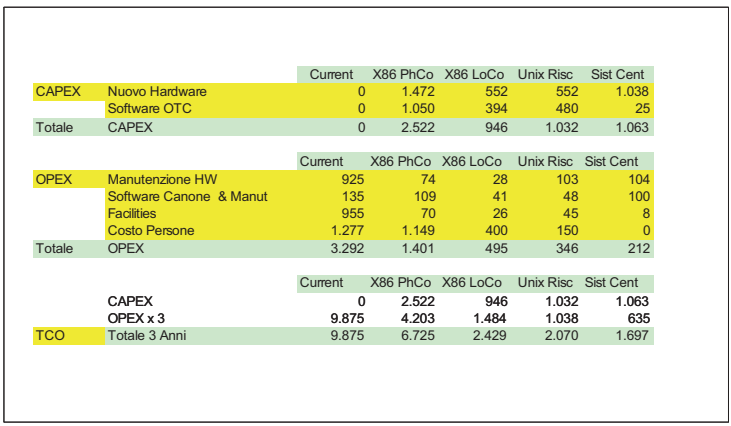


Applicazione Sportello

	Current	X86 PhCo	X86 LoCo	Unix Risc	Sist Centr
Numero Sistemi	1.534	64	24	2	1
Numero Immagini	1.534	1.534	350	10	1
Arch HW	x86	x86	x86	RISC	zArch
Op Sys	Win	Win + Vrt	Win + Vrt	Unix vrt	z/OS
Num Cpu	1.534	512	192	32	4
Imm x Syste	1	24	15	5	1

Figura 35 Caratteristiche delle Soluzioni a confronto

Le quattro soluzioni alternative sono illustrate nella figura 35. Ipotizziamo per comodità di calcolo che esse presentino funzioni uguali ed abbiano il medesimo costo di realizzazione; quindi valuteremo come componenti dei costi di acquisizione (TCA) solo i costi di Hardware e Software; nel caso dei costi Operativi (TCOp) valuteremo solo componenti di manutenzione Hardware, Canoni Software Ricorrenti, Facilities e Personale: questa ultima voce risulta essere l'elemento qualificante di ogni soluzione; infatti il costo della Infrastruttura è funzione del Numero di Immagini di Sistema, in quanto ciascuna immagine richiede delle persone per la sua gestione. La figura 36 mostra per le quattro soluzioni altenative il TCA ed il TCOp annuale, quest'ultimo confrontato con quello corrente della infrastruttura distribuita che, essendo quella attualmente in uso, presenta TCA=0. Avendo infine deciso di utilizzare un periodo di analisi di tre anni possiamo determinare il TCO (vedi figure 37, 38 e 39).



		Current	X86 PhCo	X86 LoCo	Unix Risc	Sist Cent
CAPEX	Nuovo Hardware	0	1.472	552	552	1.038
	Software OTC	0	1.050	394	480	25
Totale	CAPEX	0	2.522	946	1.032	1.063

		Current	X86 PhCo	X86 LoCo	Unix Risc	Sist Cent
OPEX	Manutenzione HW	925	74	28	103	104
	Software Canone & Manut	135	109	41	48	100
	Facilities	955	70	26	45	8
	Costo Persone	1.277	1.149	400	150	0
Totale	OPEX	3.292	1.401	495	346	212

		Current	X86 PhCo	X86 LoCo	Unix Risc	Sist Cent
	CAPEX	0	2.522	946	1.032	1.063
	OPEX x 3	9.875	4.203	1.484	1.038	635
TCO	Totale 3 Anni	9.875	6.725	2.429	2.070	1.697

Figura 36 Prospetto Costi Applicazione Sportello

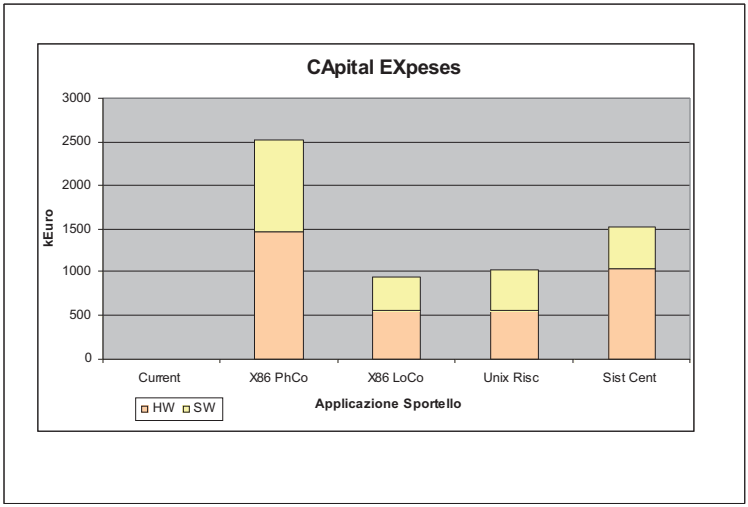


Figura 37 Costi di Acquisizione Applicazione Sportello

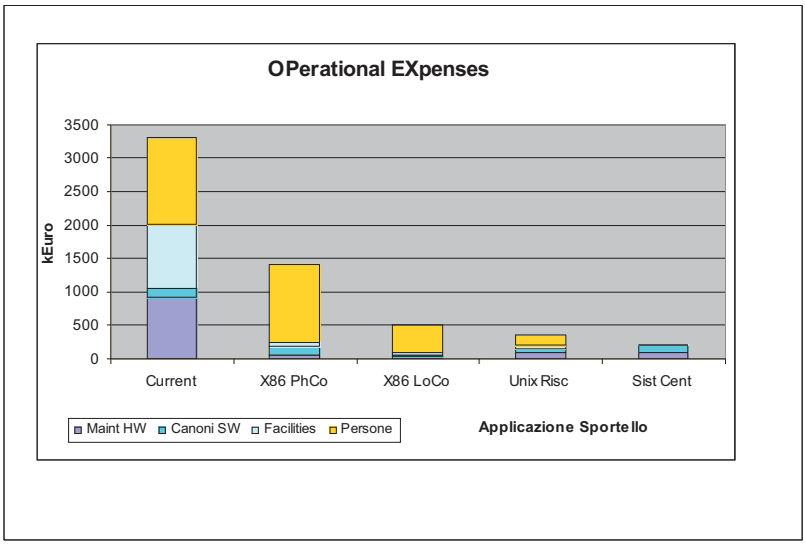


Figura 38 Costi Ricorrenti Applicazione Sportello

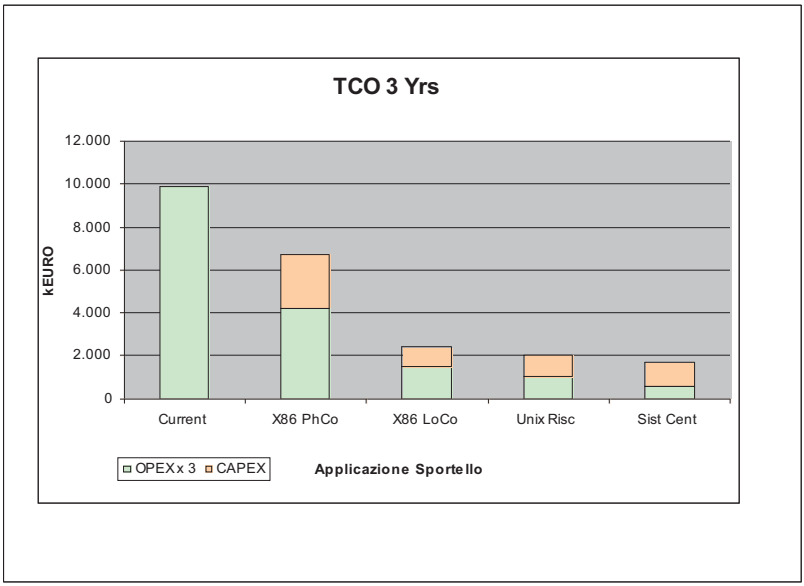


Figura 39 TCO Applicazione Sportello (3 Anni)

A questo punto siamo in grado di valutare la migliore soluzione dal punto di vista dei Costi per un arco temporale di tre anni. La figura 40 mostra l'andamento nel tempo della spesa delle quattro soluzioni considerate in relazione all'andamento dei costi correnti (rappresentato dalla linea blu)[29].

Dal grafico appare che qualsiasi soluzione considerata rappresenta un miglioramento rispetto all'andamento corrente e che il ROI risulta praticamente immediato.

Le soluzioni più convenienti risultano essere quelle che propongono un consolidamento applicativo con riduzione di numero di Immagini di Sistema (ad esempio ad una sola immagine nel caso del Sistema Centrale).

[29] Per comodidà didattica l'andamento annuale della Spesa è considerato lineare, non cambierebbe nulla nel metodo se si considerassero variazioni dovute a fenomeni inflattivi.

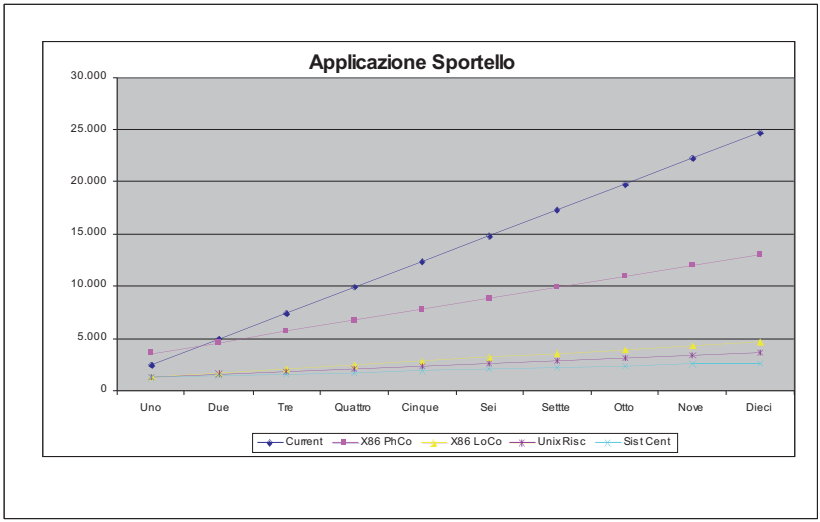


Figura 40 ROI Applicazione Sportello

4.0.15 Conclusioni

Abbiamo cercato di fornire criteri oggettivi ed il più possibile quantitativi per la selezione della Piattaforma, abbiamo introdotto i Temi dei Costi, della Tecnologia Abilitante e della Ottimizzazione della Infrastruttura Informatica (ITRO).

I processi di ITRO e di Platform Selection descritti si presentano complessi e non sempre forniscono risultati univoci: in particolare i risultati dipenderanno dai driver scelti e dal metodo usato. Abbiamo cercato di dimostrare la necessità di una metodologia con cui affrontare il problema del confronto: si è visto come il tema dei Costi possa essere legato al metodo denominato TCO e quanto sia tuttora complesso il problema della definizione di Sistemi Equivalenti. Abbiamo visto infine come la Server Consolidation possa essere considerata uno dei primi e fondamentali passi dell'ottimizzazione e come la Tecnologia della Virtualizzazione possa rappresentare un abilitatore e facilitatore del Processo. Un altro abilitatore può essere rappresentato dal Sistema Operativo Linux.

Ciò vuole dimostrare che il Sistema Centrale (Mainframe) offre un sofisticato modello di Virtualizzazione; esso consente di consolidare molti serventi e pone i primi necessari passi per l'ottimizzazione mediante la riduzione dei costi. Esso quindi rappresenta in molti casi la piattaforma migliore ed ideale per l'esecuzione di molte Applicazioni.

5.0 La famiglia dei Mainframe

Per famiglia si intende un insieme di calcolatori con la stessa architettura, quindi compatibili dal punto di vista del software sviluppato, ma con caratteristiche di prestazioni e di prezzo differenti.

Figura 1 La Famiglia del Mainframe nel 2010

La famiglia di mainframe si è evoluta nel tempo dal punto di vista delle caratteristiche hardware; tutte le famiglie sono contraddistinte da una sigla commerciale ed ogni famiglia comprende un gran numero di modelli di diversa potenza e configurazione. Ogni nuova famiglia ha conservato una caratteristica molto importante: garantire la compatibilità binaria con la precedente.

La compatibilità binaria permette ad ogni programma scritto in base alle regole della z/Architecture di essere eseguito su qualunque elaboratore compatibile senza modifiche né al source né al codice eseguibile. Ciò implica che tutti gli elaboratori "compatibili" devono eseguire tutte le istruzioni previste dalla z/Architecture con gli stessi risultati, indipendentemente dalla realizzazione tecnologica del processore (TCM, CMOS…).
La conseguenza più importante di questo è che la quasi totalità degli utenti mainframe è in grado ancor oggi di eseguire ed usare programmi compilati nel 1964 senza alcuna modifica.

Qui di seguito si riporta una breve sintesi delle principali caratteristiche delle più importanti famiglie di elaboratori IBM mainframe.

System/360

Inizialmente IBM aveva due linee separate di computer, una per il calcolo scientifico ed un'altra per quello commerciale. Si pensò di unificarle e nel 1964 uscì il primo elaboratore della linea IBM System/360 (figura 2), che aveva un addressing-mode a 24bit con una capacità totale pari a 16MB di memoria indirizzabile, ed usava la rappresentazione EBCDIC invece che ASCII. La fortuna di questa generazione di elaboratori fu la facilità di conversione da un modello all'altro, cioè l'introduzione di quel concetto di "compatibilità" espresso prima. IBM System/360 non usava la memoria virtuale.

Il design team del Sistema IBM 360 ha ricevuto dal Presidente R. Reagan il premio "National Medal of Technology".

Figura 2 S/360

System/370

Nel 1970 fu introdotta la generazione /370 (vedi figura 3), evoluzione della precedente, con raffreddamento ad acqua. Con questi elaboratori venne introdotta una nuova modalità di calcolo, il multiprocessing, cioè la possibilità di condividere il carico di lavoro su più processori. Venne anche introdotto il concetto di "virtual storage", cioè

ogni processo indirizza la propria area virtuale di memoria che viene opportunamente mappata sulla memoria reale. Alla fine degli anni '70, la serie /370 viene suddivisa in "large" 30XX series (3031, 3033, 3081, 3084) e "midrange" 43XX series (4321, 4331, 4341…) in concorrenza con i calcolatori DEC VAX ed altri simili minicomputers.

Figura 3 S/370

System/370 XA

Nel 1983 la serie "large" 30XX fu migliorata con l'introduzione dell'Architettura XA, la cui principale innovazione era la possibilità di indirizzare fino a 2GB di memoria grazie all'addressing-mode a 31bit. Nel 1985 nacque il concetto di Logical Partitioning, LPAR, cioè la possibilità di suddividere i sistemi in due o più entità logiche separate ma in grado di condividere le risorse di sistema.

System/390

Introdotta nel 1990, aveva a disposizione la nuova Architettura ESA/390, con un indirizzamento misto a 24 o 31bit, canali più veloci (ESCON) e più numerosi, maggior numero di processori e un set di istruzioni molto ampio. In questa generazione fu introdotto per la prima volta un modello CMOS, il 9221. A seguire, nel 1994, arrivarono modelli esclusivamente CMOS, i 9672 R1, raffreddati ad aria.

E' importante capire la rivoluzione circuitale che fece passare dai modelli 90xx (in tecnologia ECL, raffreddati ad acqua) ai modelli 9672 Gx (in tecnologia CMOS, raffreddati ad aria). La tecnologia ECL non poteva scendere sotto certe dimensioni dei circuiti per cui si aveva bisogno di centinaia di chip per ottenere un processore. Questi

chip venivano assemblati in Thermal Conduction Module (TCM), dove i chips erano montati in moduli di ceramica all'interno dei quali c'erano i necessari collegamenti in rame. A loro volta questi TCM erano assemblati in TCM board (50cm x 50cm; un peso di decine di Kg); un TCM board era un processore. Il problema maggiore di questa tecnologia era il grande calore generato che necessitava di complessi sistemi di dissipazione ad acqua, che rendeva notevole la dimensione degli elaboratori.

Già dal 1982, IBM stava lavorando ad una nuova tecnologia chiamata Complementary Metal Oxide Semiconductor, CMOS, che offre il grande pregio di dissipare basse quantità di calore. Inoltre, tutta la circuiteria di un singolo processore bipolare, che richiedeva circa 400 chips bipolari, con ben 4 TCM, può ora essere rimpiazzata con 4 chips CMOS ed un solo MCM). I primi modelli erano sicuramente meno potenti dei precedenti modelli 90XX e per ovviare a questo limite venne data la possibilità di accoppiare più elaboratori e costituire un'entità logica comune, un cluster, nella quale la potenza a disposizione fosse la sommatoria delle varie entità fisiche. Questo è il concetto alla base del Parallel Sysplex. Negli ultimi modelli del S/390 era presente, per la prima volta, il formato delle istruzioni IEEE floating point.

La figura 4 mostra l'evoluzione della potenza dei processori CMOS mainframe. Per fare un confronto con la tecnologia ECL si consideri che il più potente processore ECL aveva un ciclo base attorno ai 6 nsec. Il primo processore CMOS (R1) aveva un ciclo base di 18 nsec. Significativamente meno potente del processore ECL. Di qui la necessità del clustering per eguagliare la potenza ECL. Ma le possibilità di sviluppo della tecnologia CMOS erano praticamente illimitate e in 14 anni (1994 – 2008) si è passati da un ciclo base di 18 nsec a un ciclo base di 0.227 nsec. In termini di potenza di calcolo praticamente utilizzabile il processore z10 (il modello CMOS più recente) è circa 15 volte più potente dell'ultimo modello ECL (9021 711).

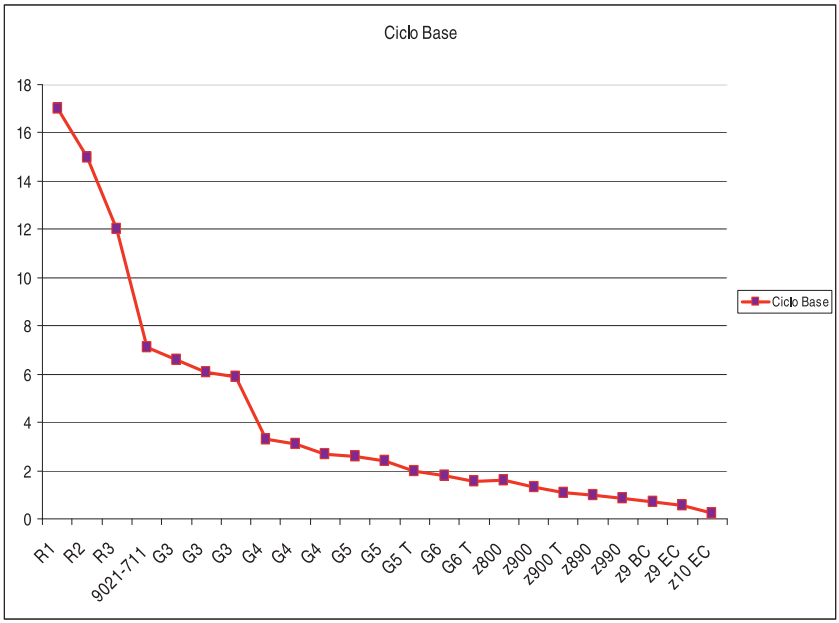


Figura 4 Evoluzione del ciclo base dei processori CMOS

zSeries & System z

L'inizio del nuovo secolo ha visto l'avvento di una nuova generazione di elaboratori, definiti zSeries, dove la z è l'acronimo di "zero downtime" e mette in evidenza la principale qualità dei mainframe, che è appunto la loro altissima affidabilità, sia hardware che software.

Viene introdotta anche una nuova architettura, la z/Architecture, la cui novità più importante è l'indirizzamento a 64bit, pur conservando la possibilità di indirizzare a 31 o 24bit, in nome della ormai famosa compatibilità.

Altra grande innovazione è la possibilità di ospitare il sistema operativo Linux, in modalità nativa all'interno di una partizione logica (LPAR), o come ospite del software di virtualizzazione z/VM.

Appendice A: I sistemi ibridi - una possibile evoluzione dei sistemi mainframe

[a cura di Marita Prassolo]

A 46 anni dalla sua prima installazione, il Mainframe gioca un ruolo centrale e vitale nei centri elaborativi delle aziende che richiedono:

- obiettivi di servizio elevati in termini di performance;
- disponibilità delle applicazioni ai numerosi utenti collegati;
- sicurezza.

Le sue caratteristiche tecnologiche e funzionali sono considerate infatti un punto di riferimento difficilmente imitabile per analoghi sviluppi nelle altre piattaforme.

Se da un punto di vista tecnologico il mainframe si rinnova ciclicamente attraverso l'adozione di novità nel disegno dei processori e di funzioni che consentono una interazione sempre più stretta con i sistemi operativi e il middleware installato, la sfida dei prossimi anni è in gran parte legata alla capacità dei Centri Elaborazione Dati di gestire applicazioni e infrastrutture sempre più complesse con tempi di reazione veloci alle richieste di cambiamenti.

Per comprendere le future evoluzioni è necessario prima di tutto analizzare quali sono i principali trend tecnologici che prevedibilmente caratterizzeranno i prossimi anni e quali sono le sfide che il mercato dell'informatica proporrà ai fornitori di tecnologia.

I trend tecnologici

Secondo le opinioni dei maggiori analisti di settore e le più recenti indagini di mercato, nel prossimo futuro l'informatica dovrà affrontare grandi sfide per soddisfare importanti esigenze di mercato quali, ad esempio:

- semplificare o meglio razionalizzare l'infrastruttura IT per migliorarne le prestazioni e per controllare la crescita dei costi di gestione che oggi avviene a ritmi inaccettabili;
- aumentare la "business resilience" cioè la capacità dell'infrastruttura informatica di fornire servizi anche in presenza di fenomeni potenzialmente catastrofici;
- aumentare l'efficacia degli apparati di sicurezza a protezione dell'infrastruttura per permetterne l'operatività secondo normative sempre più stringenti;
- migliorare la "capacità di risposta" dell'infrastruttura informatica, ovvero la capacità di adattarsi velocemente e possibilmente senza bisogno di interventi umani, a nuove condizioni di operatività;
- aumentare la flessibilità dell'infrastruttura, ovvero la capacità di supportare nuovi processi di business, senza creare vincoli per il cambiamento;
- fornire e supportare nuovi modelli applicativi che permettano di "modernizzare" le applicazioni esistenti usando le nuove tecnologie di settore (per esempio la SOA) ma conservando quanto di valido ancora esiste nelle applicazioni correnti;
- supportare i nuovi modelli applicativi emergenti.

Tutti i punti indicati possono essere considerati come facenti parte di una esigenza basilare: semplificare la gestione per consentire una crescita ordinata e coordinata dei vari componenti dell'infrastruttura.

Negli anni infatti il disegno delle infrastrutture informatiche ha visto l'inserimento di piattaforme e server di molteplici tecnologie con l'obiettivo di specializzare e/o creare isole elaborative per applicazioni o servizi.

Nella figura 1 è raffigurata, in modo semplificato, una infrastruttura tecnologica a servizo di una specifica applicazione. La complessità e la varietà di punti di interconnesione rende la gestione di queste infrastrutture onerosa e molte volte inefficente.

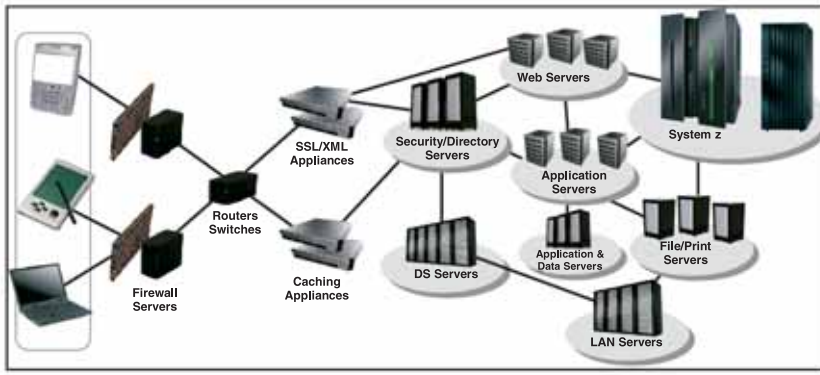


Figura 1 Infrastruttura multi-layer

L'adozione di capacità elaborative differenti è legata a molti fattori tra cui l'emergere di modelli applicativi nuovi che necessitano una forte interazione con servizi e funzioni specifiche di sistemi operativi o piattaforme diverse.

Infatti, lasciando da parte gli ambiti applicativi più specializzati tipici del mondo del calcolo intensivo e dei supercomputer, possiamo concentrarci sui modelli applicativi emergenti in un campo tradizionale dell'informatica quale il modello OLTP/DB. Questo modello applicativo, che è alla base dell'informatica moderna (si pensi alle decine di miliardi di transazioni al giorno gestite oggi da CICS ed IMS nel mondo) è da sempre caratterizzato da elevatissimi volumi di traffico, tempi di risposta cosiddetti sub-second ed enormi requisiti di sicurezza, integrità e disponibilità.

Tutto questo richiede transaction monitor di altissimo livello quali appunto il CICS e l'IMS, tecniche di programmazione avanzate per non appesantire l'esecuzione delle singole transazioni e per supportare elevati volumi, grande integrazione con i database manager (ad esempio il DB2) per un accesso ai dati rapido, efficiente e sicuro, ed una avanzata capacità di integrazione con le applicazioni batch per completare l'intero flusso applicativo nei tempi richiesti.

Con queste premesse emergono esigenze applicative sempre nuove e sempre più complesse quali:

- integrare le attuali transazioni con componenti applicativi real time ad alto contenuto di calcolo elementare e/o analitico (si pensi per esempio alle opportunità di online risk management per le applicazioni finanziarie o alle ricerche online su grandi masse di dati per applicazioni nel campo della medicina);

- utilizzare le vaste possibilità offerte dalla cosiddetta digital convergence per permettere la gestione contemporanea di dati, immagini, voce, animazioni, ecc.;

- accedere in modo rapido ed efficace a enormi quantità di dati strutturati e non strutturati (si pensi alle più moderne applicazioni di streaming computing e ai progetti in corso nel campo della sicurezza);

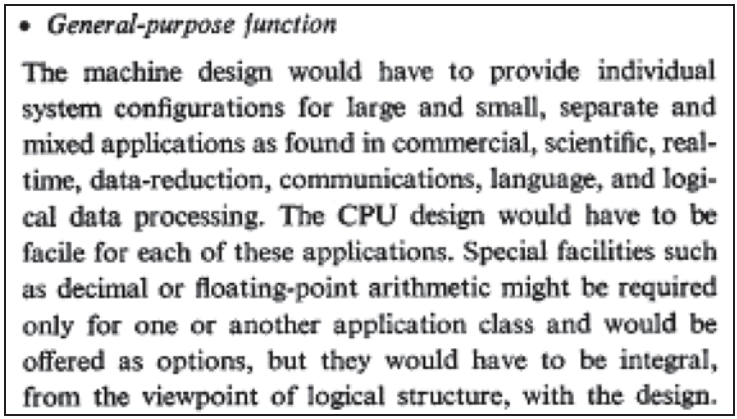
- sviluppare applicazioni di tipo "sociale" (community driven computing) con utilizzo di tecniche sofisticate di real time collaboration, gaming, ecc.

A fronte di tutto ciò è lecito chiedersi quale sia l'architettura di sistema e l'infrastruttura informatica più adatta. In particolare è importante chiedersi se sarà ancora possibile accogliere in futuro tutte queste esigenze applicative in un'unico disegno architetturale di tipo general purpose quale il mainframe e mantenere così tutti i vantaggi associati a questa tecnologia. L'alternativa è adottare architetture special purpose per eseguire particolari tipi di calcolo, rischiando però di introdurre insostenibili livelli di complessità nell'infrastruttura. Una soluzione intermedia è costituita da i sistemi ibridi. Per affrontare questo tema è opportuno ricordare alcuni elementi di differenziazione fra sistemi general purpose e sistemi special purpose.

I sistemi general purpose

Si dicono general purpose quei sistemi progettati per eseguire in modo efficiente applicazioni di qualunque tipo. La definizione è molto generica e serve in pratica solo per differenziare tali sistemi dai sistemi special purpose, che sono progettati per eseguire solo applicazioni di un tipo particolare. Nel tempo il concetto di general purpose è stato esteso per includere la capacità di eseguire contemporaneamente un numero elevato di applicazioni. Il mainframe è l'esempio tipico di sistema general purpose, in quanto è in grado di gestire contemporaneamente e con alti livelli di efficienza un numero elevato di applicazioni di tipo batch, interattivo e transazionale, di tipo commerciale e scientifico, con un vasto apparato di funzioni di controllo e di gestione fornite dal software di sistema. Il sistema IBM S/360, l'archetipo del mainframe, è definito fin dall'origine (1964) come sistema "general purpose" ed il nome stesso ne indica un possibile utilizzo su tutto lo spettro applicativo immaginabile, "a 360 gradi" appunto.

La definizione in figura 2 è presa da "The Architecture of the IBM System /360, G.M.Amdhal, G.A.Blaauw, F.P.Brooks, IBM J.Res.Dvlp, vol 44, n.1/2, January 2000" dei sistemi general purpose:



• *General-purpose function*

The machine design would have to provide individual system configurations for large and small, separate and mixed applications as found in commercial, scientific, real-time, data-reduction, communications, language, and logical data processing. The CPU design would have to be facile for each of these applications. Special facilities such as decimal or floating-point arithmetic might be required only for one or another application class and would be offered as options, but they would have to be integral, from the viewpoint of logical structure, with the design.

Figura 2 Definizione di Sistema General Purpose

Questa caratterizzazione del sistema S/360 come sistema general purpose è riflessa in molte delle decisioni architetturali prese allora quali: il formato e la lunghezza delle istruzioni, la presenza di istruzioni logiche per permettere la manipolazione di qualunque configurazione di bit e anche dei singoli bit, il meccanismo degli interrupt, i meccanismi di protezione e la gestione dell'apparato di I/O. Nel tempo la dotazione tecnologica dei mainframe è andata gradualmente aumentando, fornendo funzioni sempre più avanzate ed essenziali alla natura general purpose del sistema quali ad esempio:

- il supporto di diversi linguaggi di programmazione e relativi compilatori;
- le funzioni di dispatching e di resource management per soddisfare sia le esigenze della multiprogrammazione, sia le esigenze di utilizzo ottimale delle risorse;
- i principi e regole di system integrity;
- il multiprocessing per svincolare la capacità elaborativa complessiva del sistema dalla potenza massima erogabile con un singolo processore e quindi dai ritmi di evoluzione della tecnologia di base;
- la virtualizzazione per offrire la possibilità di gestire contemporaneamente ambienti elaborativi diversi; questa funzione, sviluppata negli anni '70 per rendere più efficiente il lavoro di un gran numero di programmatori di sistema, conserva ancora oggi tutto il suo valore, soprattutto nell'area del consolidamento e della ottimizzazione della gestione delle infrastrutture distribuite;
- l'enorme apparato di funzioni hardware e software a supporto della RAS (Reliability, Availability, Serviceability) per garantire i più alti livelli di disponibilità del servizio, anche in presenza di malfunzioni hardware, software o di ambiente.

I sistemi special purpose

A differenza degli elaboratori general purpose che, come abbiamo appena visto, sono in grado di gestire in modo efficiente e contemporaneamente un gran numero di ambiti applicativi diversi, i sistemi cosiddetti "special purpose" sono disegnati per eseguire in modo molto efficiente un ambito applicativo molto ristretto, spesso di un solo tipo. Un sistema special purpose è di solito caratterizzato da una piattaforma hardware a larga diffusione oppure, nei casi più particolari, da hardware disegnato appositamente per il particolare ambito applicativo (ASIC (Application Specific Integrated Circuit), chip). Il programma di controllo (sistema operativo) è anch'esso molto semplice dal punto di vista delle funzioni richieste ed è di solito scelto fra i prodotti a larga diffusione, ad esempio Linux. Il programma di controllo fornisce anche le funzioni necessarie a collegare questi sistemi al resto della infrastruttura informatica utilizzando i tradizionali protocolli standard.

Trattandosi di sistemi per l'esecuzione di un unico tipo di applicazione con programmi di controllo relativamente semplici, non sono richiesti sofisticati meccanismi di schedulazione, di controllo della multiprogrammazione e dell'utilizzo delle risorse. Analogamente, stante l'ambito applicativo ristretto, non sono richiesti sofisticati meccanismi di protezione della memoria, di isolamento degli utenti l'uno dall'altro, di sicurezza e di RAS. Le particolari caratteristiche dei sistemi special purpose, detti anche talvolta accelerators nelle versioni più semplici, permettono grandi livelli di efficienza in fase di esecuzione dei programmi con costi di acquisto relativamente bassi, elementi questi che rappresentano altrettanti punti di forza per queste tecnologie.

Fra i sistemi special purpose più noti ricordiamo: i sistemi di caching, i routers e gli switch di comunicazione, i processori per la crittografia e la compressione dei dati, i dispositivi per la gestione di applicazioni grafiche e multimediali, gli acceleratori per esecuzione di codice Java, i Firewall, i sistemi per il load balancing del traffico di rete e tutti quei sistemi detti genericamente internet appliance. Rientrano anche nella categoria degli elaboratori special purpose:

- i grandi supercomputer, quali il sistema IBM BlueGene, il supercomputer più potente del mondo, specializzato per l'esecuzione di calcoli particolarmente complessi quali la simulazione delle reazioni nucleari, il modo di replicarsi dei virus ed i processi interni legati allo sviluppo e formazione delle proteine;

- il sistema IBM Cell Broadband Engine, progettato inizialmente per l'ambiente di "gaming", ma dotato di tali capacità di calcolo da aver trovato immediatamente utilizzo in numerosi altri campi applicativi;

- la famiglia di prodotti DataPower specializzati per l'esecuzione di applicazioni XML.

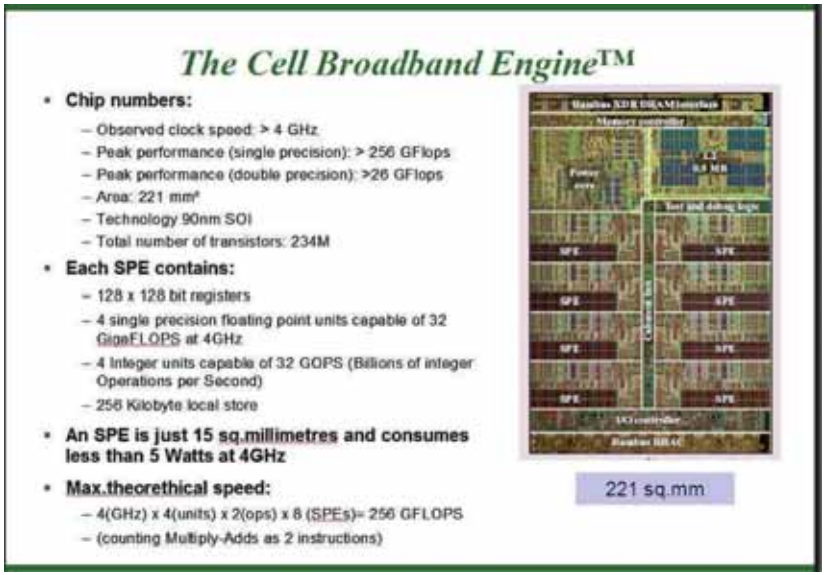


Figura 3 Cell BE Processor

I sistemi ibridi

Per tornare al tema dei sistemi ibridi, la discussione su tali sistemi nasce dalle seguenti considerazioni:

- la possibilità di aumentare le prestazioni di un processore (single thread) semplicemente aumentandone il ciclo base ha già da tempo evidenziato grossi limiti tecnologici; le soluzioni multi-core e multi-thread, pur fornendo potenze elaborative sempre crescenti, richiedono però complessi modelli di sviluppo applicativo di tipo parallelo che tardano ad emergere;

- sia i sistemi general purpose che i sistemi special purpose presentano punti di valore e indirizzano ciascuno esigenze sicuramente presenti nella maggior parte delle infrastrutture IT moderne, soprattutto di dimensioni medio grandi;

- ragionando in termini di evoluzione risulta impraticabile pensare di modificare il disegno e l'architettura dei sistemi general purpose per renderli competitivi, sia come prezzo che come prestazioni, con gli elaboratori di tipo specializzato;

- analogamente e sempre ragionando in termini evolutivi non è pensabile dotare i sistemi specializzati di tutte le funzioni tipiche dei sistemi general

purpose e pensare di poter mantenere allo stesso tempo l'efficienza di prestazioni e gli interessanti aspetti economici;

- un assetto infrastrutturale che semplicemente veda affiancati sistemi general purpose e numerosi sistemi specializzati, con livello di integrazione minimo o nullo, risulta poco praticabile sulla base dell'esperienza già oggi maturata con le infrastrutture di tipo distribuito, che vedono un incremento vertiginoso dei costi totali di gestione al crescere della molteplicità dei sistemi da gestire individualmente.

Una possibile risposta a questa complessa serie di considerazioni è oggi rappresentata dai sistemi ibridi, ovvero da sistemi costituiti da una combinazione integrata di elaboratori eterogeneii, di tipo mainframe o comunque general purpose, con elaboratori di tipo special purpose per eseguire applicazioni di tipo tradizionale e di tipo emergente, mantenendo allo stesso tempo l'immagine di sistema unico.

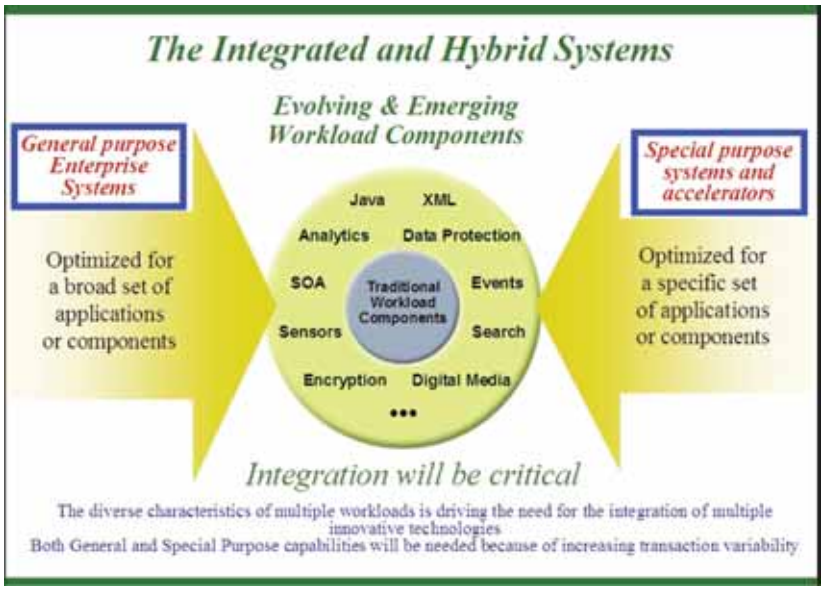


Figura 4 Sistema Ibrido Integrato

Il livello di integrazione fra i componenti di un sistema ibrido può essere di vario tipo, passando da soluzioni di tipo "loosely coupled" (interconnessione attraverso rete di comunicazione utilizzando protocolli più o meno standard) o integrazioni più strette (tightly coupled) ove l'interconnessione è a livello hardware e la velocità di comunicazione è ovviamente più elevata. In questo senso si parla anche di acceleratori o co-processori "attached" o "integrated". La realizzazione di sistemi funzionalmente avanzati mediante l'integrazione di sistemi eterogenei non è un discorso nuovo e soluzioni di tipo ibrido sono state spesso disponibili, a cominciare dalle prime configurazioni di Attached Support Processor (ASP) degli anni '70, in cui

un elaboratore S/360 veniva usato per controllare l'esecuzione dei lavori di un altro sistema S/360, di solito più potente, attraverso una connessione di tipo channel to channel operante con protocollo proprietario. L'obiettivo era di eseguire a livello di sistema operazioni complesse di schedulazione che altrimenti avrebbero dovuto essere eseguite dal personale operativo. Più di recente e per restare all'interno del mondo mainframe, ricordiamo le diverse soluzioni di integrazione di sistemi special purpose e di accelerators già presenti oggi negli elaboratori System z quali:

- i processori specializzati presenti all'interno del chip per le funzioni crittografiche e la funzione di compressione dei dati;

- i processori specializzati derivati dai motori standard per fornire supporto a particolari funzioni di sistema (motori ICF) o per ottimizzare i costi complessivi in presenza di particolari ambiti applicativi quali Linux (motori IFL), programmi Java (motori zAAP) e tipologie particolari di accesso ai dati (motori zIIP).

Infine ricordiamo gli ultimi importanti sviluppi di integrazione di sistemi eterogenei, cioè integrazione del sistema z con il processore Cell Broadband Engine e integrazione del sistema z con i sistemi DataPower.

Il processore Cell Broadband Engine (Cell BE), annunciato da IBM nel 2005 come risultato di un progetto congiunto Sony, Toshiba e IBM, iniziato nel 2001, è già di per se un sistema ibrido, in quanto combina in un unico chip di dimensioni contenute un motore general-purpose di architettura Power e di modesta capacità elaborativa con numerosi (attualmente 8) processori specializzati che impiegano modelli di esecuzione SIMD (Single Instruction Multiple Data) secondo la classificazione di Flynn (Flynn, M., Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, pp. 948, 1972) per l'esecuzione di carichi applicativi ad alto contenuto computazionale, quali i calcoli analitici, le applicazioni multimediali ed il calcolo vettoriale. Il programma di controllo è basato su Linux.

Le prime realizzazioni di integrazione fra i sistemi z ed il processore Cell BE sono del tipo loosely-coupled con collegamento via rete e protocollo TCP/IP per supportare una nuova generazione di applicazioni internet 3D.

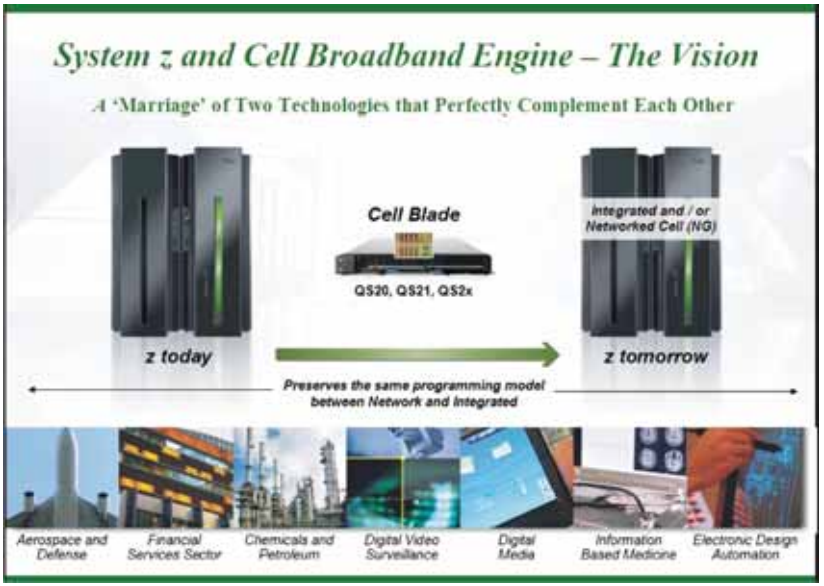


Figura 5 Il primo esempio di Sistema Ibrido

Un discorso molto simile può essere fatto per l'integrazione del sistema z con le cosiddette IBM WebSphere DataPower SOA appliances. DataPower, originariamente un costruttore americano di dispositivi di rete, è stato acquisito da IBM nel 2005, diventando una divisione prodotti che produce appliances XML per l'elaborazione di messaggi XML e per la più generica trasformazione di messaggi per le comunicazioni server-to-server.

Tali appliances usano hardware ASIC ed un programma di controllo basato su Linux. Queste soluzioni permettono già oggi di aumentare le prestazioni, la sicurezza e la gestione di applicazioni che fanno uso di standard XML con configurazioni ibride loosely-coupled.

Anche per questo tipo di servizio è ipotizzabile nel futuro una soluzione ibrida integrata. Ciò permetterà di associare le importanti funzionalità DataPower con la qualità di servizio dei sistemi z. In termini più generali, è possibile prevedere per il futuro la disponibilità di sistemi ibridi che integrino in modo più o meno stretto funzioni di tipo general purpose con funzioni di tipo special purpose.

Le caratteristiche principali di queste soluzioni potranno essere:

- uno strato di hardware costituito da un numero variabile di sistemi di tipo sia general purpose che special purpose affiancati dai vari tipi di co-processor ed accelerator che l'industria informatica renderà disponibili;

- una serie di dispositivi di interconnessione per consentire comunicazioni veloci e sicure fra i vari componenti hardware e per consentire l'accesso ai dispositivi esterni;

- uno strato di virtualizzazione (Hypervisor) che mascheri all'esterno la molteplicità anche architetturale dei componenti hardware e fornisca importanti funzioni quali il provisioning on demand;

- una serie di istanze di ambienti elaborativi, caratterizzate ciascuna dal proprio sistema operativo e dalle proprie applicazioni e con la possibilità di condividere i dati;

- uno strato di software di controllo per la gestione dell'intero sistema che mantenga nei confronti dell'utilizzatore esterno l'immagine del sistema unico. Stante la criticità, soprattutto per le prestazioni di questo elemento di controllo, è prevedibile che questa funzionalità venga realizzata in firmware.

I punti di forza di una evoluzione della piattaforma z verso soluzioni di tipo ibrido sono:

- la possibilità di eseguire sulla stessa piattaforma workload di tipo tradizionale e workload di tipo emergente, rimanendo sempre in un ambiente operativo con le caratteristiche della piattaforma z;

- la possibilità di utilizzare nuove tecnologie e mantenere una visione unica di sistema, con le caratteristiche tipiche del mondo mainframe, per quanto riguarda le tecniche di gestione e l'accesso ai dati;

- la possibilità di far crescere le dimensioni dell'infrastruttura informatica al crescere delle esigenze applicative, senza per questo incidere in modo determinante sui costi di gestione.

Con queste premesse il 22 luglio 2010 IBM ha annunciato l'elaboratore IBM zEnterprise System. zEnterprise è il primo elaboratore ibrido disponibile sul mercato e ingloba la potenza elaborativa della piattaforma z, quella di processori di tecnologia Power 7 e, nel prossimo futuro, di tecnologia Intel.

Il componente IBM zEnterprise Unified Resource Manager invece avrà il compito di controllare e gestire l'intero sistema.

Ogni componente citato mette a disposizione degli utenti funzioni con lo scopo di rendere più semplice ed ordinata la crescita applicativa richiesta da un mondo in continua evoluzione ed espansione. In particolare nell'annuncio sono citati:

- IBM zEnterprise 196
- IBM zEnterprise zBX
- IBM zEnterprise Unified Resource Manager

Nella figura 6 è rappresentato il contenuto dell'annuncio del 22 luglio 2010.

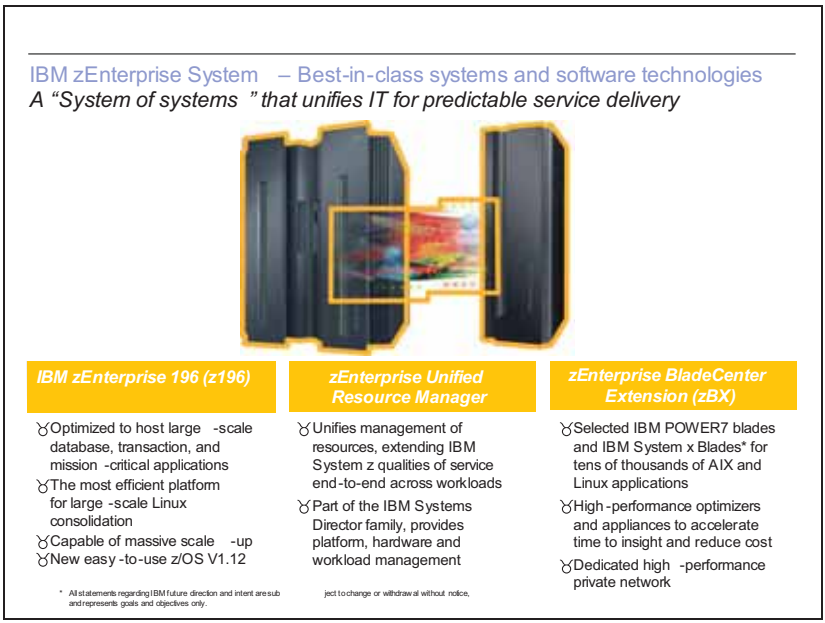


Figura 6 Principali componenti di IBM zEnterprise System

Il primo sistema ibrido: IBM zEnterprise System

Come anticipato il nuovo elaboratore ha componenti differenti: nei paragrafi seguenti sono descritti alcuni elementi del contenuto e delle funzioni di questo nuovo sistema.

IBM zEnterprise 196 (z196)

E' la componente legata all'evoluzione della piattaforma z. Come i suoi predecessori, IBM System z10 Enterprise Class, è composto da 4 book, ciascuno con processori che, con configurazioni personalizzate, possono gestire sistemi operativi diversi quali:

- z/OS
- z/VM
- z/VSE
- z/TPF
- zLinux

Il numero di processori disponibili agli utenti raggiunge il numero di 80 a 5,2GHz configurabili come nel passato. É a tutti gli effetti il processore commerciale più potente sul mercato con un incremento complessivo del 60% rispetto al suo predecessore. Nella figura 7 è possibile notare l'evoluzione rispetto alle famiglie precedenti in termini di potenza.

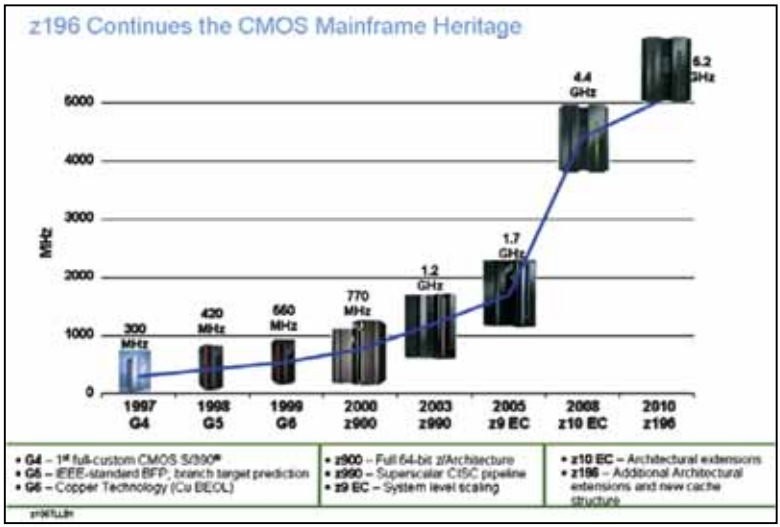


Figura 7 Evoluzione della tecnologia nella piattaforma z

La crescita è stata notevole anche in altri componenti quali la memoria e le dimensioni della cache interna dei processori (che è stata completamente ridisegnata in termini di quantità e livelli interni).

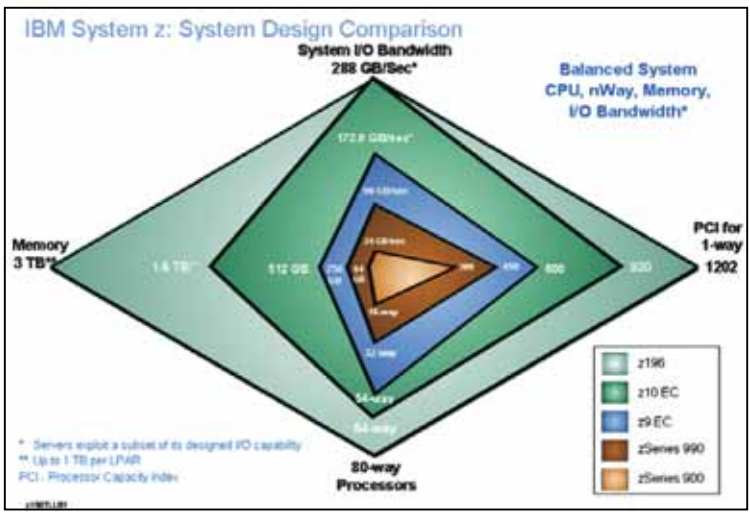


Figura 8 Risorse disponibili negli elaboratori System z

Nella figura 8 vengono visivamente indicate le principali risorse disponibili agli utenti del sistema nelle ultime generazioni di elaboratori: è interessante notare il raddoppio della memoria configurabile sul sistema z196 pari oggi a 3TB.

IBM zEnterprise zBX (zBX)

Nei 4 rack collegabili ad un z196 e condivisi con altri zEnterprise sono integrate tutte le componenti di una infrastruttura complessa. Infatti sono previsti dispositivi di rete e verso l'I/O per collegare i processori Power 7 (ed in futuro Intel) disponibili su blade (lame) e gestiti tramite IBM zEnterprise Unified Resource Manager.

I processori possono essere utilizzati in due modalità distinte:

1. Ottimizzatori (strettamente correlati con i sistemi operativi z/OS ed il middleware). Sono già annunciate due tipologie:
    a. IBM Smart Analytics Optimizer
    b. Data Power (acceleratore XML- Statement of Direction)
2. IBM Blades: in cui sistemi operative AIX (per P7) e Linux (per Intel) utilizzano un ambiente virtualizzato gestito direttamente dall' Unified Resource Manager

Nella figura 9 è rappresentato schematicamente un complesso elaborativo IBM zEnterprise System con i sistemi operativi e gli ottimizzatori disponibili attualmente sull'hardware zBX.

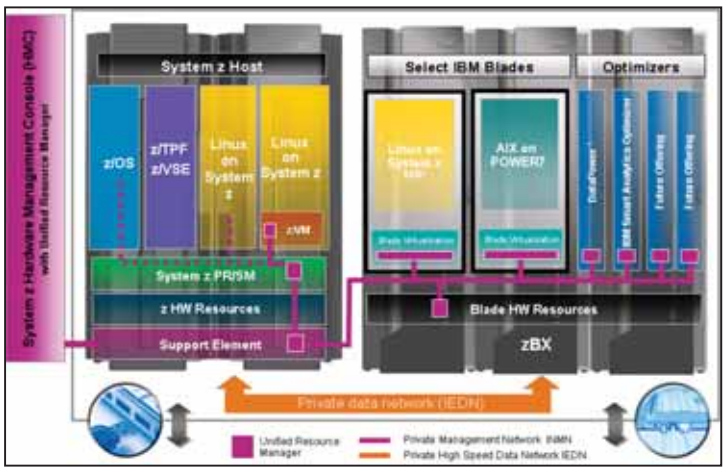


Figura 9 Schema logico di IBM zEnterprise System

IBM zEnterprise Unified Resource Manager (URM)

Questa componente ha il compito di gestire e controllare un insieme di sistemi operativi e componenti hardware. Ad un Unified Resource Manager si possono connettere un massimo di 8 IBM zEnterprise System componendo in tal modo un

Ensemble. Gli Ensemble sono collegati tra loro da reti TCP/IP dedicate alla trasmissione di dati tra sistemi operativi o di comandi gestionali per i componenti hardware e software. La configurazione massima di un Ensemble è costituita da 8 z196 e 32 rack zBX.

Unified Resource Manager gestisce la virtualizzazione del complesso elaborativo e fornisce l'interfaccia verso gli hypervisor garantendo un unico punto di controllo e di gestione. E' suo compito garantire un monitoraggio integrato e facilitare la problem determination degli eventuali malfunzionamenti.

Grazie alle funzioni di questo componente sarà possibile realizzare un controllo end-to-end della transazione e determinare (o lasciare determinare) le esigenze in termini di azioni/risorse per raggiungere i livelli di servizio pianificati.

Nella figura 10 sono visualizzate le funzioni dell'Unified Resource Manager che, una volta pienamente implementate e rese disponibili, semplificheranno la gestione delle infrastrutture necessarie ai sistemi informativi delle aziende.

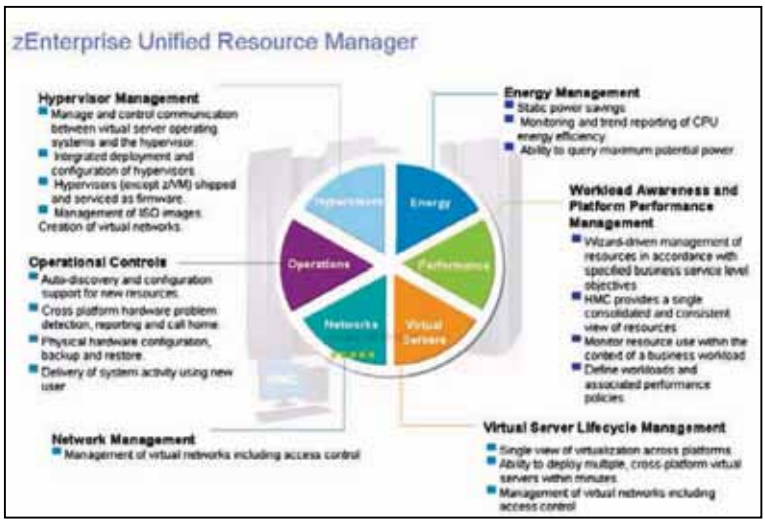


Figura 10 Funzioni di IBM zEnterprise Unified Resource Manager

Il Futuro è gia Presente

Con l'annuncio del 22 luglio quindi è iniziata una nuova fase per i Centri Elaborativi delle Aziende poichè il sistema zEnterprise introduce nuovi elementi da considerare nella fase di scelta delle infrastrutture necessarie per i livelli di servzio desiderati e per le crescite previste.

Nella figura 11 è ben rappresentata la crescita di potenza che un sistema ibrido potrà erogare utilizzando componenti di piattaforme e tecnologie diverse ma integrate tra loro.

Come già precisato, gli aumenti della potenza della piattaforma z si ottengono attraverso la maggiore capacità di ogni singolo processore e il maggior numero di processori utilizzabili o da una singola immagine di sistema operativo o all'interno dello stesso box.

Con un sistema elaborativo ibrido la potenza disponibile alle applicazioni ed alle infrastrutture informatiche cresce notevolmente, mantenendo la visione (dal punto di vista gestionale ed infrastrutturale) di un singolo sistema.

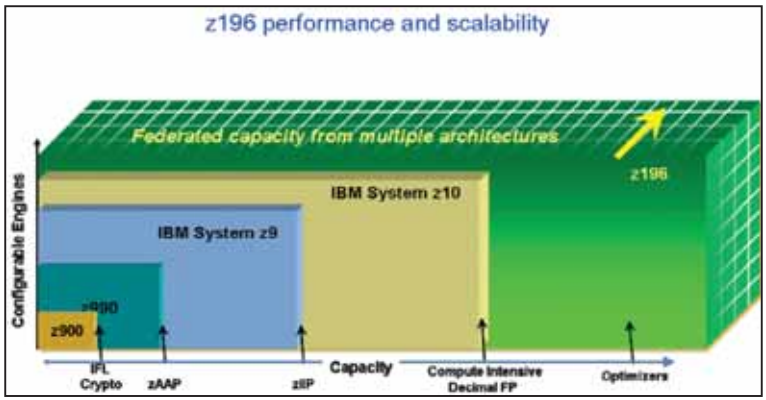


Figura 11 La nuova dimensione di potenza dei sistemi ibridi

Questa potenza disponibile sarà gestita e monitorata come una unica entità garantendo quindi una semplicità di utilizzo che consentirà ad applicazioni sempre più interconnesse tra loro e con funzioni sempre più complesse di interagire nel modo migliore.

La figura 12 offre una visione di questo nuovo ambiente mostrando come sarà possibile creare Cluster di applicazioni che utilizzano piattaforme hardware diverse pur avendo le stesse intefacce gestionali.

Osservando con attenzione infatti un disegno architetturale di una soluzione mainframe ibrida, è possibile notare come tutto il complesso elaborativo è attraversato da codice con il compito di federare tra loro sistemi e risorse diverse: il colloquio con i software sarà garantito da agent sparsi nei diversi livelli. Inoltre la virtualizzazione consentirà la creazione di cluster o pool di sistemi con lo scopo di bilanciare il workload ed ottenere performance ottimali.

Da una infrastruttura così integrata possono beneficiare tutte le applicazioni oggi presenti nei Centri Elaborazione Dati poichè la semplificazione e la condivisone significano performance migliori e gestione meno onerosa.

Inoltre le nuove applicazioni ed il middleware potranno sfruttare possibilità offerte dalla presenza così strettamente cooperante di piattaforme tra loro diverse. Sono state

citate nei precedenti paragrafi applicazioni che utilizzano già, in ambiente mainframe, una architettura ibrida (ad esempio la crittografia), ma il nuovo annuncio lascia intravedere sviluppi ulteriori.

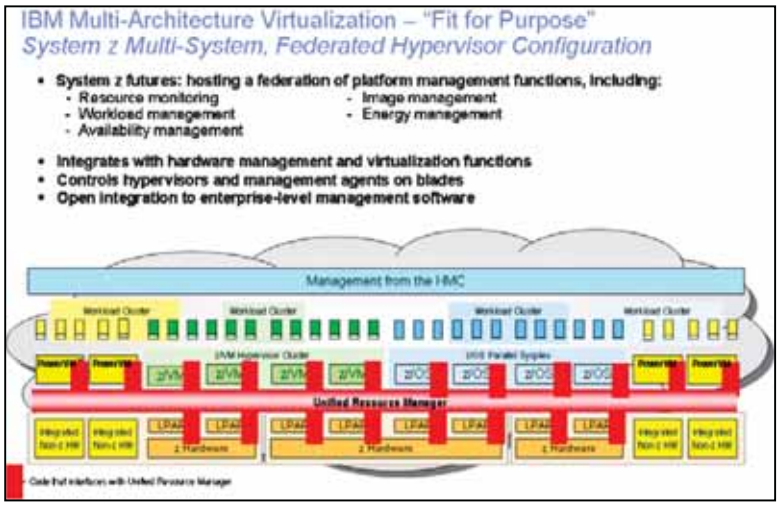


Figura 12 Esempio di sistema federato

Una delle prime aree che potranno beneficiare di un sistema mainframe ibrido sarà quella legata alle applicazioni di Business Intelligence and Analytics. Utilizzando una nuova funzione del DB2 V9 alcune richieste saranno inviate, in modo trasparente alle applicazioni stesse, su processori specializzati che eseguono velocemente le azioni richieste avendo i dati necessari in memoria.

Questi processori sono fisicamente operativi in un IBM zEnterprise zBX collegato mediante link ad alte prestazioni ed integrato gestionalmente tramite l'Unified Resource Manager.

Figura 13 Esempio di sistema ibrido

Con una tale realizzazione sarà possibile mantenere più facilmente i dati necessari al Data Warehouse allineati con i dati operativi ed avere prestazioni elevate per le richieste più complesse necessarie agli utenti di queste applicazioni.

Appendice B: Sicurezza e Crittografia dei Mainframe

[a cura di Massimo Leoni]

Introduzione

Prima di affrontare la sicurezza nel *System z* nei suoi vari aspetti, è necessario fare una panoramica del problema della sicurezza nell'IT e individuarne gli elementi portanti e i cardini di riferimento. Innanzi tutto ci si deve porre la domanda, anche se sembrerà pleonastica: che cos'è la sicurezza? Il termine sicurezza rappresenta generalmente una forma d'assicurazione, con un costo associato, a fronte di un valore che si vuole proteggere, per raggiungere uno stato di confidenza per "sentirsi" sicuri, a fronte di:

- Rischi individuati a fronte di eventi accidentali o deliberati
- Normative cui conformarsi

La **Tabella 1** riporta una possibile classificazione dei tipi di rischi associati ad un ambiente IT[1].

Umano		**Ambientale**
Volontario	**Accidentale**	
Intercettazioni	*Errori e omissioni*	*Terremoto*
Modifica dell'informazione	*Cancellazione di file*	*Fulmine*
Hacking	*Instradamento errato*	*Allagamento*
Virus (Codice Maligno)	*Incidenti fisici*	*Incendio*
Furto		

Tabella 1

Dalla tipologia di eventi riportati possiamo rilevare un ulteriore raggruppamento in eventi che coinvolgono la sfera fisica ed altri invece che si esauriscono nel dominio logico. Gli eventi che afferiscono al mondo fisico sono studiati dalle discipline del "*Disaster Recovery*" e della "*High Availability*" che sviluppano teorie e soluzioni in questi campi. Noi invece ci occuperemo della sicurezza logica, cioè dello studio e della definizione di soluzioni che riguardano eventi volontari o accidentali che possono compromettere l'integrità del sistema di elaborazione e/o dei dati. Procediamo ora a sviluppare un metodo e un modello che permetta di tracciare una strada per la costruzione e la comprensione di una soluzione di sicurezza. Innanzitutto è necessario classificare gli "*asset*", cioè le risorse (dati, programmi, comunicazioni, ecc. per quanto riguarda la sicurezza logica) che sono necessarie all'operatività aziendale e che sono

[1] IT – Information Technology

alla base dell'integrità dei processi dell'impresa. Classificare, in questo contesto, significa identificare le risorse ed attribuire loro un valore aziendale. Un valore che deve potere essere "assicurato".

Ad un primo livello di astrazione questa "assicurazione" si traduce nel definire delle politiche che appunto mirino a mitigare il rischio, difendendo gli *asset* da minacce o adeguando comportamenti nel rispetto di normative richieste dal contesto di mercato in cui opera l'azienda (es. Legge sulla Privacy, Sarbanes Oxley Act ecc.). Ora che abbiamo le politiche, si pone la questione di come farle applicare. Ci sono due alternative, l'utilizzo di:

1. **Disciplina -** Vengono diramate e messe a conoscenza dei dipendenti delle norme comportamentali e organizzative a protezione degli *asset* e volte anche al controllo del loro accesso;
2. **Meccanismi -** Vengono utilizzate delle tecnologie e dei meccanismi che permettono di costringere i comportamenti dei dipendenti in aderenza alle politiche aziendali.

I meccanismi sono la modalità che, dal punto di vista della tecnologia ed in questa trattazione, ci interessano maggiormente. Quindi vediamo ora come collocare e classificare i vari meccanismi per poterli impiegare profittevolmente nella sicurizzazione di ambienti *System z*.

Partiamo con una classificazione dei domini della sicurezza logica, che per comodità baseremo su quella proposta nello standard ISO 7498-2 (rappresentato graficamente in **Figura 1**).

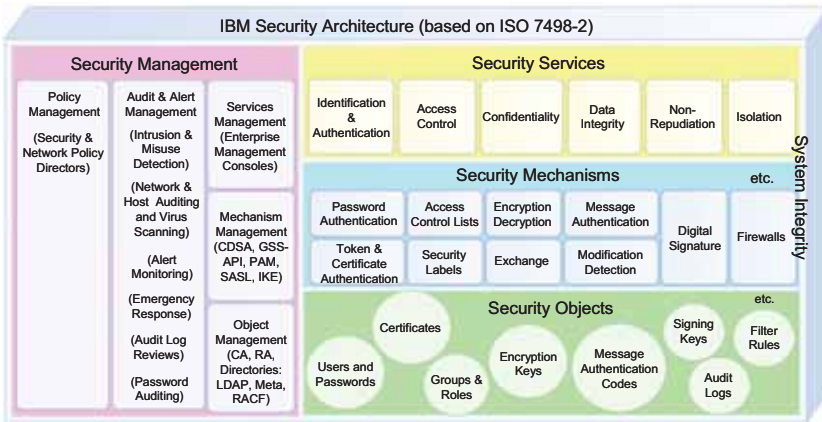


Figura 1 IBM Security Architecture

Questa classificazione permette di catalogare gli ambiti di applicazione dei meccanismi di sicurezza nelle seguenti categorie:

- Autenticazione

- Autorizzazione e controllo degli accessi
- Riservatezza/Confidenzialità
- Auditabilità e Non Ripudio

che andiamo a collocare sul modello riportato in **Figura 2**.

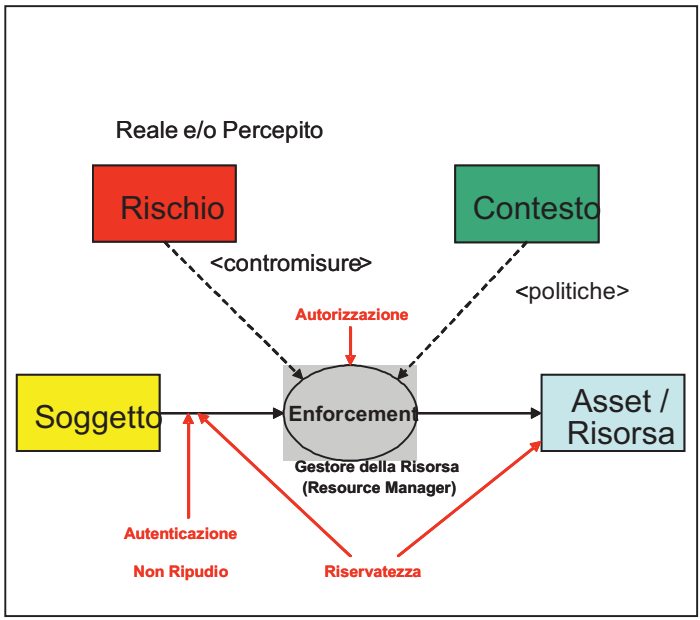


Figura 2 Catalogazione dei meccanismi di sicurezza

Nel modello si evidenzia come un soggetto, che accede ad un asset o ad una risorsa tramite un mediatore (gestore delle risorse o *resource manager*), deve essere in qualche modo autenticato. Inoltre anche il mezzo di comunicazione utilizzato per lo scambio di informazioni tra il soggetto e il gestore delle risorse può essere protetto (riservatezza e non ripudio) così come i dati stessi (*Asset*/Risorsa). La decisione di lasciar accedere il soggetto alla risorsa viene presa dal gestore delle risorse in base a delle politiche derivate dal contesto e da meccanismi (contromisure) che invece sono correlate al livello di rischio legato all'accesso di queste risorse.

Tutto questo, per poter rispondere in modo coerente alle aspettative di chi implementa soluzioni di sicurezza, si deve fondare obbligatoriamente sulla integrità del proprio contenitore (Piattaforma Hardware, Sistema Operativo, ecc.), in modo da garantire che le politiche e la semantica dei meccanismi siano rispettate.

Il paragrafo precedente suona come l'enunciazione di un assioma. Tutta la sicurezza che si va a costruire si fonda sull'integrità. Ma come si valuta o si verifica la bontà di una soluzione di sicurezza o le affermazioni di integrità fornite dai vari produttori di Hardware e Software ? La risposta è la certificazione. Per garantire che una soluzione, un meccanismo, un sistema hardware o software rispetti le specifiche di disegno, le pratiche riconosciute di realizzazione e sia progettato per resistere ad attacchi conosciuti (e sconosciuti), esso deve essere certificato. Le metodologie e gli standard di certificazione più rilevanti per quanto riguarda la sicurezza sono:

- ISO/IEC 17799 (ex BS 7799 – *British Standard 7799*)
- ISO/IEC 15408 (conosciuto come *Common Criteria* o *CC*, i livelli di validazione e certificazione sono riportati in **Tabella 2**)
- FIPS 140 (*Federal Information Processing Standard* – conosciuto soprattutto per la certificazione di Hardware Crittografico)

Common Criteria Assurance levels
EAL1: *Functionally Tested*
EAL2: *Structurally Tested*
EAL3: *Methodically Tested and Checked*
EAL4: *Methodically Designed, Tested and Rev iewed*
EAL5: *Semiformally Designed and Tested*
EAL6: *Semiformally Verified Design and Tested*
EAL7: *Formally Verified Design and Tested*

Tabella 2

L'integrità del System z

Iniziamo ora a percorrere e a collocare, all'interno della piattaforma *System z* e z/OS, i concetti che abbiamo appena introdotto. Analizziamo per primo l'integrità del sistema hardware. Come sappiamo il *System z* è un elaboratore che, attraverso un ipervisore a livello firmware, il PR/SM, può essere virtualizzato, cioè le risorse fisiche della macchina possono essere condivise da più immagini di sistema operativo. In questo caso è necessario garantire che ogni immagine di sistema operativo ospitato in una partizione logica della macchina (LPAR) sia equivalente ad un elaboratore fisico isolato. Per questo il *System z z10 EC e BC* è certificato, rispetto a questa caratteristica, con i *Common Criteria (CC)* a livello EAL5.

Un'altra importante caratteristica hardware del System z è la possibilità di proteggere l'accesso alla memoria reale con delle chiavi di memoria o "*Storage Key*". Ad ogni

blocco di 4KB (pagina) di memoria è associata una chiave (*storage key*) che permette di controllare a livello hardware l'accesso in scrittura e/o lettura al contenuto della pagina. Se un programma ha associato la chiave di memoria 0, significa che può accedere a qualsiasi pagina di memoria reale (di solito questa è la chiave del sistema operativo). Altre chiavi sono assegnate dal sistema operativo ai sottosistemi principali (es. VTAM, JES2, IMS ecc.) che utilizzano a loro volta questo meccanismo per proteggere le proprie aree di memoria da accessi non autorizzati (per un errore o intenzionali) da parte di programmi utente.

Passiamo ora all'integrità del software. L'integrità nel sistema operativo z/OS è un aspetto fondamentale del disegno della piattaforma. Questa caratteristica è talmente importante che viene supportata da una forte dichiarazione da parte di IBM attraverso l' *IBM STATEMENT OF MVS (z/OS) SYSTEM INTEGRITY*, che afferma l'impegno di IBM ad apportare correzioni (a qualsiasi costo) al sistema z/OS per malfunzionamenti che inficino l'integrità del sistema stesso: deve essere impossibile per un programma non autorizzato da un meccanismo sotto il controllo del cliente:

1. aggirare o disabilitare la protezione di *store* (scrittura in memoria) o *fetch* (lettura dalla memoria);
2. accedere ad una risorsa di Sistema protetta da password o da RACF[2];
3. ottenere il controllo in uno stato autorizzato da programma, cioè essere in supervisor state con una chiave di protezione inferiore a otto (8), o essere autorizzato in *Authorized Program Facility* (APF).

Oltre a questa dichiarazione di IBM, il System z, congiuntamente con lo z/OS 1.7, è certificato con i *Common Criteria (CC)* a livello EAL4.

Autenticazione, Autorizzazione e controllo Accessi nello z/OS

Come abbiamo visto in Figura 2 , i meccanismi di Autenticazione e Autorizzazione (Controllo degli Accessi) sono utilizzati dal *resource manager* per prendere delle decisioni riguardo l'accesso alle risorse che controlla (ad esempio il sottosistema CICS - *resource manager* - e le sue transazioni - *resources*). Il modello generale di Figura 2 può essere ora contestualizzato nel caso dello z/OS nella figura 3:

[2] RACF – Resource Access Control Facility, è il servente di sicurezza del sistema z/OS.

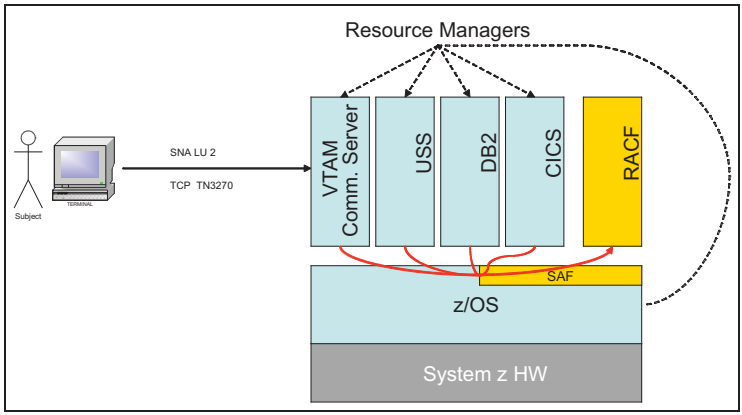


Figura 3 Modello di sicurezza dello z/OS

Vediamo ora come le varie componenti e in particolare il RACF, contribuiscono a realizzare la soluzione di sicurezza per lo z/OS.

SAF

La *System Authorization Facility* (SAF) è parte del sistema operativo z/OS e fornisce una serie di interfacce a servizi di sistema per operare l'autenticazione, l'autorizzazione e la tracciatura (*auditing* e *logging*) degli eventi di sicurezza. La SAF fornisce una modalità che si occupa di rispondere a richieste di un gestore di risorse (*Resource Manager).* L'elemento chiave della SAF è il *SAF router* (modulo ICHSFR00). Il *router* è sempre presente in una installazione di z/OS anche se RACF non è installato. Il *SAF router* rappresenta il punto comune di ingresso alle funzioni di sicurezza che possono quindi essere condivise tra prodotti e sistemi. Un gestore delle risorse (*resource manager*) può richiamare il *SAF (z/OS) router* attraverso una macro assembler (RACROUTE) come parte di un flusso che comporti decisioni di controllo di accessi (REQUEST=VERIFY) o di autenticazione di un soggetto (REQUEST=AUTH). Queste funzioni sono chiamate *control points*. Questa nuova infrastruttura di sicurezza fornisce molti benefici per la realizzazione di un sistema sicuro e promuove il suo uso da parte di programmi e componenti del sistema operativo.

RACF

RACF è l'acronimo per *Resource Access Control Facility* ed è una componente fondamentale nell'architettura di sicurezza dello z/OS. Esso incorpora al suo interno diversi elementi del modello della sicurezza, come l'autenticazione e l'identificazione degli utenti e il controllo degli accessi. RACF è quindi un fornitore di servizi alla SAF e fornisce le risposte alle richieste dei vari gestori di risorse. Per completezza di informazione dobbiamo anche dire che esistono sul mercato alternative a RACF come

security manager fornite da altri fornitori di software come CA-ACF2 e CA-TopSecret della Computer Associated.

Architettura del RACF

Dal punto di vista architetturale, RACF si avvale di una base dati (RACF database) nella quale vengono memorizzati i profili:

- degli utenti (USER);
- dei loro raggruppamenti (GROUPS);
- delle risorse da proteggere;
- dei suoi parametri di funzionamento.

Il base ai suoi algoritmi di verifica RACF, basandosi sulle informazioni contenute nella sua base dati, può fornire una risposta alle richieste ad esso pervenute dalla SAF, invocate dai resource manager o da programmi utente. Nella Figura 4 è rappresentata sinteticamente l'architettura di RACF, dove viene mostrato il flusso di una richiesta di accesso ad una risorsa all'interno di un ambiente z/OS. L'utente o il programma innanzitutto devono essere autenticati dal resource manager o dal componente di sistema operativo a cui richiedono il servizio. Risultato dell'autenticazione, che avviene richiamando appunto RACF, è la creazione da parte dello z/OS di un ACcessor Environment Element (ACEE). L'ACEE rappresenta l'identità di un utente o di un programma (Address Space o Task/Thread) ed è l'elemento che viene utilizzato in seguito per il processo di autorizzazione.

Quando un utente o un programma autenticato richiede l'accesso ad una risorsa, il resource manager chiede a RACF se l'utente è autorizzato al suo uso, secondo la modalità richiesta (es. lettura, scruttura o aggiornamento). In questa richiesta vengono passate a RACF le seguenti informazioni:

- l'ACEE dell'utente (o del programma);
- il nome della risorsa che l'utente vuole accedere;
- il tipo di accesso richiesto (es. READ, WRITE, UPDATE, ...).

RACF verifica se l'utente è autorizzato, eventualmente registra un record di audit avvalendosi della System Management Facility (SMF), e restituisce un "parere" al resource manager. Questo parere può essere di tre tipi:

- l'utente è autorizzato all'accesso (RC[3]=0);
- RACF non può prendere una decisione perchè la risorsa non è protetta da un profilo (RC=4);
- l'utente non è autorizzato all'accesso (RC=8).

[3] RC – Return Code

A questo punto il resource manager, sentito il parere di RACF, decide operativamente se garantire o meno l'accesso alla risorsa (normalmente in modo congruente alla risposta di RACF).

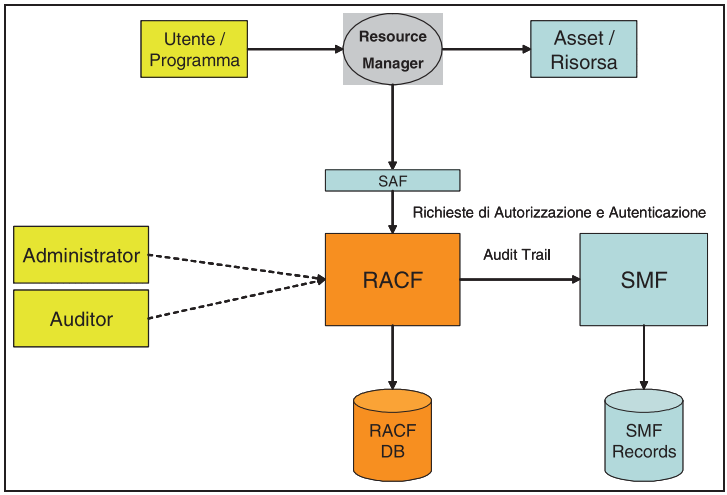


Figura 4 Architettura RACF

Identificazione e Autenticazione dei soggetti (utenti, programmi, ecc.)

L'autenticazione nello z/OS, come abbiamo visto, avviene a cura del resource manager, cioè quel programma che, attraverso un protocollo o API[4], interagisce con un utente per fornire un servizio. Se il processo di autenticazione termina positivamente, viene creato un ACEE, che come abbiamo mostrato in precedenza, è l'elemento che viene utilizzato per verificare il controllo dell'accesso alle risorse. Per poter abilitare questo meccanismo, l'amministratore di RACF deve definire i profili degli utenti (USER PROFILEs) ed eventualmente anche gli insiemi di raggruppamento degli utenti o gruppi (GROUP PROFILEs) che verranno memorizzati nel database RACF. All'interno del profilo dell'utente sono memorizzate sia informazioni di tipo informativo (es. USER DATA) sia informazioni relative a specifiche applicazioni (USS[5], CICS, IMS SEGMENTs).

A questo punto, per autenticare un utente, il resource manager deve ottenere, attraverso il protocollo di comunicazione, delle credenziali che certifichino che l'utente è quello che afferma di essere. In seguito, effettuata la verifica delle credenziali, è necessario, per la creazione dell'ACEE, associare il soggetto riconosciuto tramite le credenziali con l'utente definito all'interno del database RACF (quello che è riconosciuto a livello di sistema). Quest'ultima funzione prende il nome di "*identity*

[4] API – Application Programming Interface
[5] USS – Unix System Services

mapping" che all'interno del sistema z/OS e RACF può essere sia di tipo banale, come nel caso della user/password, nella quale la user usata corrisponde a quella registrata in RACF, sia di tipo più complesso, come il *mapping* di un soggetto di un certificato digitale X.509 con un utente RACF o utilizzando i servizi di Enterprise Identity Mapping (EIM).

Come riferimento, riportiamo di seguito alcuni esempi di credenziali:

- Utenza e Password
- Certificati digitali X.509 (SSL e TLS[6])
- RSA SecureID
- Kerberos
- ...

Nella Tabella 3 riportiamo degli esempi di abbinamento Resource Manager / Protocollo utilizzato per accedere la risorsa / Credenziali.

Resource Manager	***Protocollo/API***	***Credenziali***[7]
TSO	• TN3270 • SNA LU2	• User/Password o Passticket[8] • Certificato X.509
WebSphere/HTTP	• HTTP • RMI/IIOP	• User/Password o Passticket • Certificato X.509
JES (z/OS Batch)	• JCL	• User/Password o Passticket
CICS	• CTG[9] • SNA LU2	• User/Password o Passticket
DB2	• DRDA • JDBC	• User/Password o Passticket • Kerberos

Tabella 3

[6] SSL – Secure Socket Layer; TLS – Transport Layer Security
[7] Non tutte le credenziali elencate si applicano ai protocolli presenti nella casella adiacente
[8] Passticket – E' un meccanismo di "*one time password*" offerto da RACF
[9] CTG – CICS Transaction Gateway

Autorizzazione e controllo degli accessi

Quando un utente (o programma) è stato autenticato, può operare all'interno dello z/OS e interagire con componenti del sistema operativo o con resource manager, come il CICS o il DB2. Al fine di verificare se un utente può avere accesso ad una risorsa, è necessario definire uno specifico profilo nel database RACF. Questo profilo include i permessi di accesso alla risorsa stessa. Il resource manager passerà poi a RACF questo profilo per verificare le autorizzazioni di accesso dell'utente a quella specifica risorsa.

I profili di risorse in RACF sono identificati da una classe di appartenenza (RACF CLASS). Il profilo identifica il tipo di risorsa (es. PROGRAM) ed ha un nome univoco associato (es. per un dataset SYS1.USERS.ROSSI). Ne consegue che il resource manager (es. DB2, CICS, ...), quando richiama RACF per un parere sull'autorizzazione all'accesso, deve inserire nella richiesta di autorizzazione sia l'ACEE dell'utente sia la classe e il nome della risorsa.

Per poter prendere delle decisioni sull'accesso alla risorsa, RACF può utilizzare due schemi di autorizzazione, singolarmente o in combinazione:

1. *Discretionary Access Control* (DAC) nel quale le autorizzazioni sono esplicitamente assegnate agli utenti o ai gruppi di utenti. Queste autorizzazioni prendono il nome di *Access Control List* (ACL) e fanno parte del profilo della risorsa;
2. *Mandatory Access Control* (MAC) nel quale l'autorizzazione viene valutata in base ad attributi definiti sia a livello di profilo utente che di profilo della risorsa. Gli attributi utilizzabili per questo metodo di autenticazione in RACF sono: il SECLEVEL, le CATEGORY e le SECLABEL (la SECLABEL di fatto è una combinazione di CATEGORY e SECLEVEL).

Vediamo in maggior dettaglio i due modelli di autorizzazione.

Il modello DAC richiede che siano definiti gli utenti, i gruppi di utenti e le risorse. L'amministratore della sicurezza modificherà la ACL del profilo di una risorsa in modo tale che garantisca (o neghi) esplicitamente l'accesso alla risorsa (o alle risorse) protetta da quel profilo RACF, indicando anche il tipo di accesso che viene autorizzato (es. READ, UPDATE, ...). Per far questo l'amministratore inserisce nella ACL il nome di un utente o di un gruppo. E' buona pratica autorizzare l'accesso alle risorse ai gruppi di utenti, in questo modo l'accesso alla risorsa avviene semplicemente connettendo un utente ad un gruppo di utenti, semplificando e separando la fase di disegno del controllo degli accessi da quella di amministrazione ordinaria.

Il modello MAC, rispetto al DAC, non necessita del concetto di gruppo e l'amministratore della sicurezza di fatto garantisce un accesso implicito alle risorse, assegnando ad esse e agli utenti degli attributi (es. SECLABEL). Vediamo di chiarire meglio questo concetto con un esempio. Supponiamo di creare una tabella di classificazione utilizzando livelli di sicurezza (SECLEVEL) di classificazione dell'informazione identificati da Top Secret, Confidential e Unclassified. Ortogonalmente, definiamo delle category, che per esempio corrispondono ad unità di business come HR, Personnel, Engineering, ecc. Facendo in questo modo otterremo

una tabella come quella descritta all'interno della Figura 5. Creiamo ora dei "tesserini" (SECLABEL) associando categorie e classificazioni di sicurezza, che nel nostro caso chiamiamo:

- SECLABEL A per la combinazione (HR, Top Secret);
- SECLABEL B per le combinazioni ((HR, Top Secret), (HR, Confidential), (Personnel, Confidential)).

In questa situazione, se un utente ha un tesserino che "include" il tesserino della risorsa (primo esempio in Figura 5, allora l'accesso può essere garantito, nel caso contrario (secondo esempio in Figura 5) l'accesso viene negato.

Nella realtà, l'implementazione delle modalità di autorizzazione MAC in RACF è più complessa e articolata di quanto mostrato in quest'esempio. Inoltre la modalità MAC è alla base della cosiddetta *Multi Level Security* (MLS), che aggiunge al modello MAC il modello di sicurezza Bell-La Padula[10].

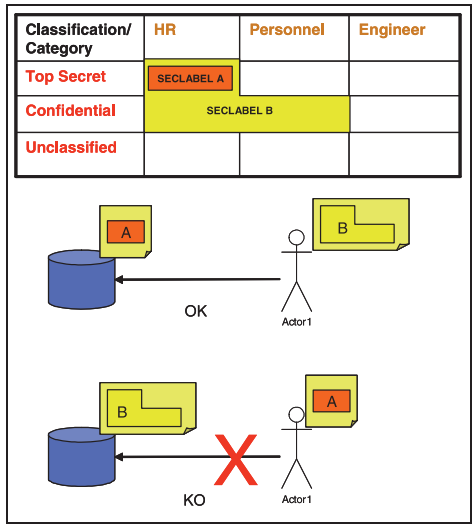


Figura 5 Esempi modello MAC

Amministrazione e Auditing

L'amministrazione ed il controllo di un ambiente protetto da RACF è in genere operata da più ruoli con autorità in ambiti amministrativi disgiunti, in modo da avere una separazione di potere che garantisce un minor rischio rispetto ad attività malevole di tipo volontario. Gli ambiti di amministrazione/gestione definiti in RACF sono:

[10] http://en.wikipedia.org/wiki/Bell-LaPadula_model

- *Amministrazione e Gestione del RACF* - è caratterizzato dall'attributo SPECIAL nel profilo RACF dell'utente. Questa tipologia di utente è in grado di definire profili (utente, gruppo ed risorse) ed impostare i parametri di funzionamento di RACF;
- *Libera operatività e accesso alle risorse del sistema* – è caratterizzato dall'attributo OPERATIONS nel profilo RACF dell'utente.
- *Impostazione e rapportistica dei parametri di auditing del sistema e delle risorse* - è caratterizzato dall'attributo AUDITOR nel profilo RACF dell'utente.

Gli amministratori si avvalgono, per amministrare RACF, delle interfacce native, TSO e ISPF, o di prodotti come IBM/CONSUL.

IBM Security Server

L'IBM Security Server è un "impacchettamento" di prodotti e componenti di sicurezza relativi alla piattaforma z/OS. I componenti che ne fanno parte, oltre a RACF, sono i seguenti:

- *DCE*[11] *Security Server* - fornisce le funzioni di un server di sicurezza OSF[12] DCE;
- *Lightweight Directory Access Protocol (LDAP) Server* - è basato su un modello client/server e fornisce accesso tramite LDAP ad una directory di tipo generico e ai dati relativi alle utenze e gruppi memorizzati nel database di RACF;
- *z/OS Firewall Technologies* (da z/OS 1.8 parte integrante di z/OS Communication Server) – è l'implementazione sia di un firewall di rete IPV4 con il suo meccanismo di controllo accessi di rete basato su filtri, sia di un server IPSEC[13] per la realizzazione di reti private virtuali (VPN) che vedono la partecipazione di istanze z/OS;
- *Network Authentication Service for z/OS* - fornisce i servizi di sicurezza di Kerberos integrati con RACF;
- *Enterprise Identity Mapping (EIM)* - sono dei servizi infrastrutturali a disposizione di applicazioni e sistemi operativi per gestire più facilmente il flusso delle identità tra sistemi e base dati utenti (*User Registry*) diverse all'interno di una soluzione d'impresa;

[11] DCE – Distributed Computing Environment. http://www.opengroup.org/dce/
[12] OSF – Open Software Foundation, now The OpenGroup http://www.opengroup.org/
[13] IPSEC – IP Security http://www.ietf.org/html.charters/OLD/ipsec-charter.html

- *PKI*[14] *Services* - fornisce i servizi di Certification Authority e di gestione del ciclo di vita dei certificati digitali X.509 (emissione e amministrazione) necessari a soluzioni di firma digitale o di autenticazione forte (es. SSL v3 o TLS Client Authentication).

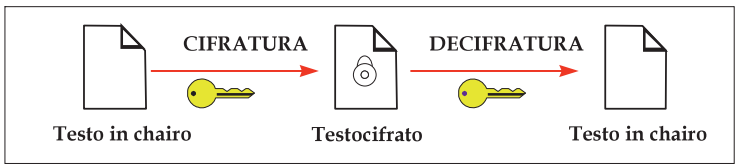
Riservatezza (Confidentiality): la crittografia

La **crittologia** è la scienza che studia i modi e le tecniche per segretare delle informazioni (crittografia) o come rivelare informazioni segretate senza esserne autorizzati (crittoanalisi). Essa si compone quindi di due branche:

- **Crittografia** (nome derivato dal greco κρυπτός kryptós "nascosto" e il verbo γράφω gráfo "scrivere") che si occupa dello studio della segretazione delle informazioni. Attualmente è diventata un ramo della teoria dell'informazione, spostandosi dall'originario ambito linguistico a quello matematico. Ron Rivest, noto crittografo e co-inventore della crittografia a chiavi pubbliche RSA[15] ha osservato che "la crittografia è relativa alla comunicazione in presenza di avversari";
- **Crittoanalisi** (nome derivato dal greco κρυπτός kryptós "nascosto" e il verbo αναλύειν analýein "sciogliere, risolvere") si occupa invece dello studio di metodi per ottenere l'informazione cifrata, senza aver accesso all'informazione segreta che è stata usata per cifrarla. Nel liguaggio comune questa pratica viene anche chiamata "*codebraking*" o "*cracking the code*" cioè "rompere" il codice.

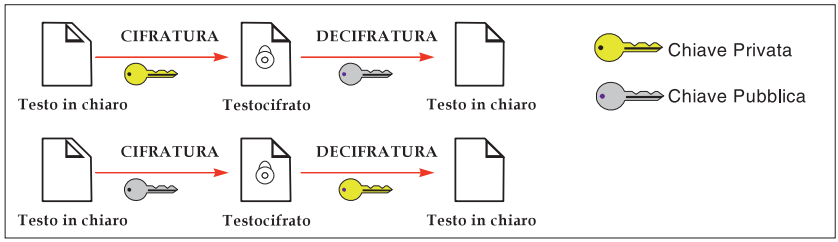
La crittografia moderna, come abbiamo accennato prima, si fonda sull'utilizzo della matematica per inventare dei metodi che, utilizzando un'informazione segreta detta chiave, permettano di segretare delle informazioni. Naturalmente, il disegno dell'algoritmo crittografico garantisce che più una chiave è "lunga" più è difficile (se non impossibile) effettuare la crittoanalisi in tempi accettabili. Attualmente possiamo classificare i metodi (e algoritmi) crittografici in tre categorie principali:

- *Crittografia a chiavi simmetriche* - questa classe di algoritmi utilizza la stessa chiave sia per cifrare che per decifrare l'informazione;

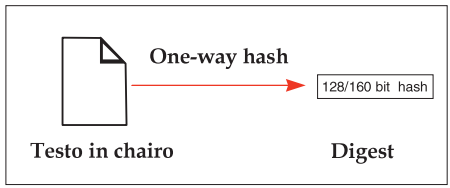


[14] PKI – Public Key Infrastructure o Infrastruttura a Chiavi Pubbliche.
[15] RSA – Rivest, Shamir e Adelman, matematici e fondatori della società RSA, proprietaria del brevetto della crittografia a chiavi pubbliche detta comunemente crittografia RSA.

- *Crittografia a chiavi asimmetriche o a chiavi pubbliche* - questa classe di algoritmi utilizza due chiavi, univocamente associate l'una all'altra. Una chiave, detta privata, è usata per cifrare ed un'altra chiave, detta pubblica, per decifrare. Vale inoltre la proprietà inversa, cioè quello che è stato cifrato con la chiave pubblica può essere decifrato unicamente con la chiave privata associata;



- *Hash o digest* - questa classe di algoritmi non necessita di una chiave per operare. Il loro scopo è quello di trasformare un'informazione di lunghezza arbitraria, nel modo più univoco possibile, in una stringa di lunghezza fissa e relativamente limitata (detta *hash* o *digest*). Queste funzioni operano in un solo senso ossia l'informazione originaria non può essere ricostruita partendo dall'*hash*.



Nella Tabella 4 sono riportati alcuni algoritmi di crittografia più comunemente usati.

Algoritmi Simmetrici	Algoritmi Asimmetrici	Algoritmi Hash
DES, Triple-DES (3DES), AES, RC4, Blowfish	RSA, DSA, ELGAMAL	RIPEMD, SHA-1, MD5

Tabella 4

Come e quando vengono utilizzati questi algoritmi e perché ? Per rispondere a questa domanda dobbiamo analizzare le caratteristiche di questi algoritmi (es. prestazioni,

livello di sicurezza offerto, lunghezza di chiave utilizzata, ecc.) e il contesto in cui vengono impiegati.

Prestazioni - sotto questo aspetto gli algoritmi simmetrici sono decisamente più efficienti di quelli asimmetrici. Quando si deve cifrare una notevole quantità di dati o effettuare la cifratura frequentemente, in genere si utilizzano algoritmi simmetrici.

Contesto - con l'avvento di Internet il contesto informatico operativo coinvolge quasi sempre (se non sempre) due sistemi che si scambiano informazioni e servizi attraverso un mezzo di comunicazione accessibile a tutti. In questo caso se, come abbiamo detto in precedenza, utilizziamo la crittografia simmetrica per cifrare la comunicazione, sorge il problema di condividere la stessa chiave tra i due interlocutori. Questo problema naturalmente cresce con il numero degli interlocutori. In nostro aiuto però viene la cifratura a chiavi pubbliche, che può essere impiegata nella cifratura di pochi dati, come le chiavi, e in attività eseguite non frequentemente, come lo scambio di chiavi. In effetti le soluzioni di comunicazione con cifratura, come l'SSL, il TLS e IPSEC, impiegano una combinazione di algoritmi di cifratura simmetrica e asimmetrica per raggiungere l'obiettivo di realizzare una comunicazione sicura efficiente ed efficace su internet.

Le funzioni di hash, in combinazione con cifratura simmetrica o asimmetrica, sono invece utilizzate per garantire l'integrità delle informazioni tramite tecniche di MAC e di firma digitale o per derivare chiavi crittografiche e password da un testo od una frase.

Prima di passare all'architettura crittografica del System z, spendiamo ancora qualche parola su alcuni concetti che ci torneranno utili a breve. Come abbiamo visto nei paragrafi precedenti, l'elemento portante della sicurezza è la chiave di cifratura. Essa è un elemento di controllo e non deve essere rivelata a persone non autorizzate a conoscerla; se ciò avvenisse, l'informazione protetta dalla chiave sarebbe esposta a manipolazioni, alterazioni e così via. Come però ben sappiamo, all'interno di un elaboratore, per poter eseguire un programma (in questo caso l'operazione di cifratura o decifratura) è necessario che i dati (in questo caso anche la chiave) siano in memoria: questo significa che la chiave di cifratura è "in chiaro" e potrebbe essere letta da chi è in grado di leggere nella memoria dell'elaboratore.

Per ovviare a questa situazione è necessario realizzare un ambiente di elaborazione protetto, sia logicamente che fisicamente, nel quale operare la cifratura o la decifratura delle informazioni. Solamente in questo ambiente, detto *Hardware (o Host) Security Module* (HSM), la chiave viene messa in chiaro. In questo caso parliamo di chiave di cifratura "protetta", che si contrappone alla cifratura con chiave "in chiaro". Gli HSM possono prendere forma di dispositivo esterno all'elaboratore o di scheda interna (es. PCI/PCIX) e sono acceduti tramite delle API come quelle definite nello standard RSA PKCS#11 (conosciuto anche come *Cryptoki*).

L'architettura del sottosistema crittografico

L'architettura del sistema crittografico si basa sulla *IBM Common Cryptographic Architecture* (IBM CCA), che definisce i componenti e le interfacce applicative (API) nonché le modalità di interfacciamento ai componenti crittografici di sistema. La caratteristica principale della CCA, oltre alla definizione dei servizi crittografici e a

come invocarli, risiede nella "architettura" delle chiavi di cifratura. Lo scopo è quello di avere un disegno di riferimento per la cifratura con chiavi "protette" avvalendosi di un HSM. La principale classificazione delle chiavi comprende due categorie principali:

1. *Master Key* (MK) - sono chiavi che vengono inserite nell'HSM (e non possono poi essere più estratte) che servono a proteggere le chiavi operative. Le Master Key cifrano le chiavi operative generate o inserite all'interno dell'HSM in modo tale che possano essere memorizzate con sicurezza esternamente all'HSM, per esempio su un database. In questo modo si è sicuri che l'utilizzo di quelle chiavi può essere fatto unicamente conoscendo la Master Key di protezione, e quindi, per quanto abbiamo appena detto, solo all'interno dell'HSM. In genere le Master Key sono chiavi a cifratura simmetrica di tipo 3DES o AES;
2. *Operational Keys* (OPKey) o chiavi operative - sono le chiavi utilizzate per effettuare la cifratura, possono essere sia di tipo simmetrico che asimmetrico ed in genere sono specializzate per un particolare tipo di cifratura (legata quindi ai servizi della CCA). Ad esempio una chiave può essere utilizzata solo per generare PIN[16] bancari, un'altra per cifrare delle chiavi, un'altra ancora per effettuare firme digitali. Questo controllo sull'utilizzo è a cura dell'HSM tramite l'utilizzo di *Control Vector*, cioè delle maschere che si combinano con la chiave operativa per ottenere l'effettiva chiave di cifratura.

L'hardware: CPACF e schede crittografiche CEX3C

Nel System z le "*facility*" crittografiche, ovvero il complesso dei componenti crittografici, è caratterizzato da due elementi hardware:

- CPACF (CP[17] Assist for Cryptographic Functions) - è un componente hardware del processore (CP) che può essere utilizzato attraverso istruzioni (assembler) dell'architettura z. Il CPACF è un'estensione dell'Instruction Set della z/Architecture presente in ogni processore z e la sua esecuzione è sincrona rispetto al flusso di esecuzione delle istruzioni del programma. Esso è molto veloce nell'esecuzione di istruzioni crittografiche ed opera sia con chiavi "in chiaro" sia con chiavi protette (*wrapped key*). Gli algoritmi sono solo di cifratura simmetrica, come il 3DES e l'AES, di hashing, come lo SHA-1, e di generazione di numeri casuali (PRNG[18]);
- Il coprocessore IBM 4764 - è un "*embedded system*" alloggiato su una scheda PCIX ed implementa un HSM per le piattaforme hardware IBM. Sul System z il coprocessore IBM 4764 è chiamato *Crypto Express3 Coprocessors* (CEX3C) che al suo interno include due IBM 4764. Il CEX3C è una scheda PCI che, al contrario del CPACF, è acceduta in asincrono e attraverso un

[16] PIN – Personal Identification Number. Il codice segreto utilizzato dalla carte bancarie come ad esempio il Bancomat per autorizzare delle transazioni (es. Prelievo di contante).
[17] CP – Central Processor. Il processore del System z.
[18] PRNG – Pseudo Random Number Generator

sottosistema crittografico ICSF[19]. Non è possibile accedere con delle API native ai servizi CEX3C ma bisogna transitare attraverso i servizi ICSF. Esso può essere configurato sull'elaboratore in due modalità:

- come acceleratore crittografico (CEX3A) per eseguire velocemente operazioni di cifratura asimmetrica con chiavi "in chiaro";
- Come co-processore crittografico (CEX3C) per eseguire funzioni crittografiche simmetriche, asimmetriche e di tipo bancario / finanziario con chiavi "protette".

Esiste nell'architettura System z un altro componente hardware (opzionale) che indirizza la gestione sicura delle chiavi. Questo componente è una workstation, la *Trusted Key Entry* (TKE) *workstation,* che appunto è in grado di comunicare, in modo protetto, le chiavi crittografiche ai cripto-processori, senza farle transitare in chiaro nella memoria dell'elaboratore (vedi figura 6).

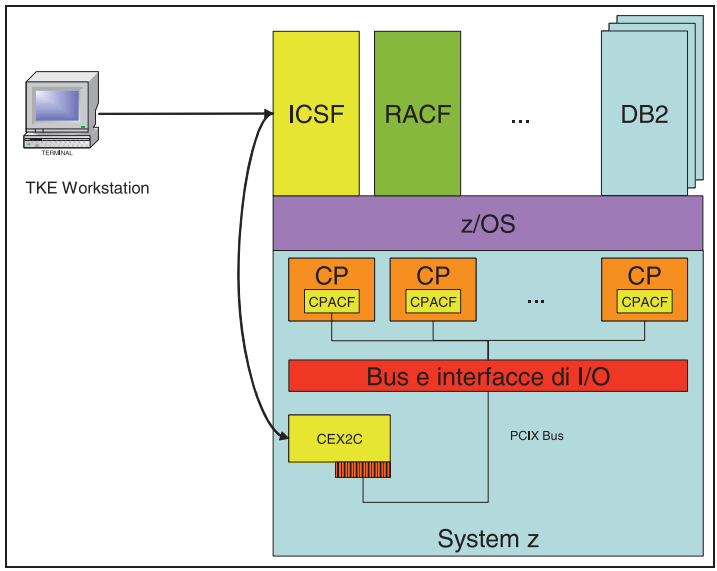


Figura 6 Architettura dell'ambiente crittografico nel System z

[19] ICSF – Integrated Cryptographic System Facility

Il software: ICSF e applicazioni/framework crittografici

Il gestore delle risorse crittografiche, cioè i servizi (algoritmi) e le chiavi, in z/OS è l'Integrated Cryptographic Service Facility (ICSF). Il compito di ICSF è quello di offrire un'interfaccia applicativa (API) in conformità alla IBM CCA per accedere ai servizi crittografici forniti dall'hardware. Dal punto di vista della sicurezza e del controllo degli accessi, ICSF si appoggia a RACF per prendere queste decisioni, in particolare si avvale di due classi di risorse, la CSFSERV e la CSFKEYS, che proteggono rispettivamente l'accesso ai servizi e alle chiavi crittografiche secondo il modello di sicurezza RACF implementato. ICSF è quindi utilizzato sia da diversi "framework" di sicurezza, come z/OS System SSL o l'implementazione OCSF (CDSA) [20], sia direttamente da vari linguaggi di programmazione, quali C/C++, COBOL, FORTRAN, ecc., come rappresentato in Figura 7.

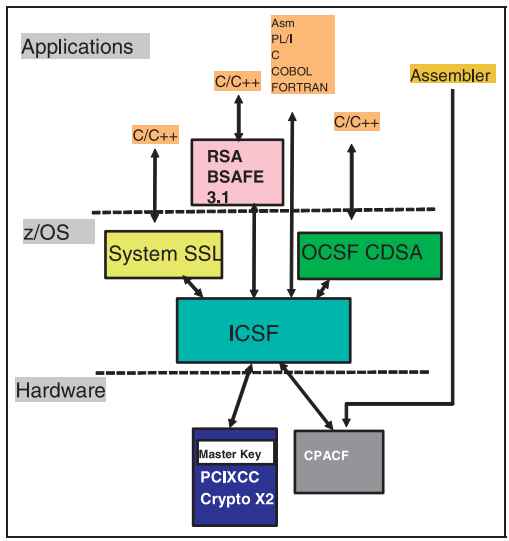


Figura 7 ICSF – Architettura

Un'altra funzione di ICSF, oltre ai servizi crittografici, è quella di offrire un ambiente per la gestione delle chiavi crittografiche. Per far questo – come da Figura 8- ICSF si avvale di tre basi dati (dataset VSAM[21]):

- Una chiamata CKDS[22], che contiene chiavi operative simmetriche;

[20] OCSF - Open Cryptographic Service Facility è una implementazione della Intel Common Data Security Architecture (CDSA)
[21] VSAM – Virtual Storage Access Method
[22] CKDS – Cryptographic Key Data Set

- Una chiamata PKDS[23], che contiene chiavi operative asimmetriche;
- Una chiamata TKDS[24], che contiene token PKCS#11.

Le chiavi registrate nei dataset sono cifrate con le master key, che a loro volta sono mantenute in sicurezza all'interno del criptoprocessore. Il CKDS è protetto dalla Master Key Simmetrica (SYM-MK) mentre il PKDS è protetto dalla Master Key Asimmetrica (ASYM-MK). E' da notare che, a dispetto del nome, entrambe sono chiavi per cifratura simmetrica Triple DES (3DES). L'inserimento delle Master Key nel criptoprocessore può avvenire attraverso le interfacce amministrative di ICSF o, in modalità più sicura, utilizzando la workstation TKE.

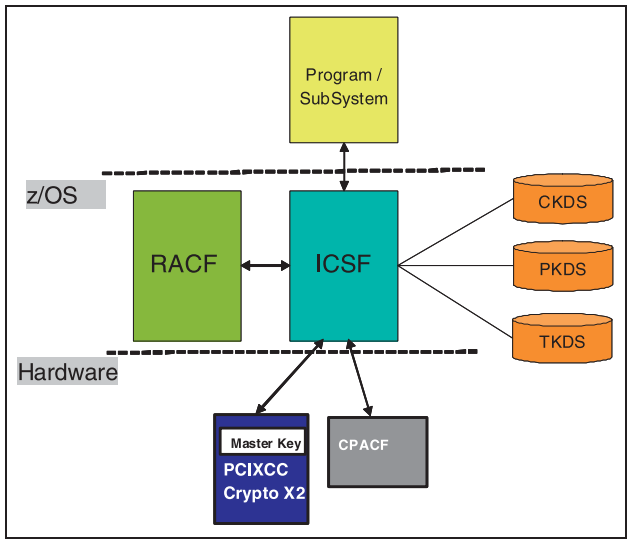


Figura 8 ICSF – Gestione chiavi

ICSF ha inoltre introdotto il supporto, a livello di algoritmi software, alla crittografia basata su curve ellittiche (Elliptical Curve Cryptography (ECC)). Nella versione attuale ICSF è in aderenza sia alla RFC4869 "Suite B Cryptographic Suites for IPSEC" sia al supporto di nuovi algoritmi con chiavi in chiaro:

- Galois/Counter Mode encryption per AES (GCM) ;
- Elliptic Curve Diffie-Hellman key derivation (ECDH);
- Elliptic Curve Digital Signature Algorithm (ECDSA) e HMAC.

[23] PKDS – Public Key Algorithm Data Set
[24] TKDS – Token Data Set

Linux

Anche Linux, se ospitato su di un elaboratore System z in modo nativo o come ospite dell' ipervisore z/VM [25], è in grado di sfruttare le funzionalità hardware della piattaforma. In questo ambiente non esiste un sottosistema come l'ICSF dello z/OS ma viene fornito un apposito device driver tramite il quale ricevono l'accesso ai cripto processori *framework* o API di sicurezza come l'OpenSSL, PKCS#11 e l'implementazione SSL di IBM, il GSKit. La Figura 9 mostra l'architettura crittografica (Hardware e Software) su zLinux.

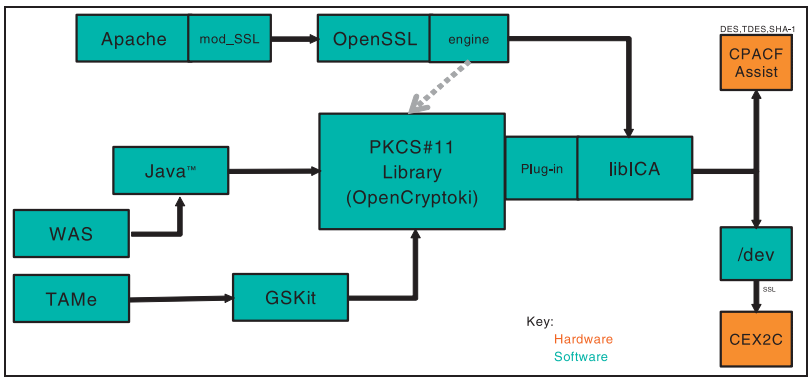


Figura 9 Architettura crittografica su zLinux

[25] z/VM – System z Virtual Machines

Appendice C: I sottosistemi Storage

[a cura di Stefano Ricci]

I Sottosistemi di memorizzazione delle informazioni sono una parte essenziale delle piattaforme server. Verranno descritti quei sottosistemi in uso nelle installazioni mainframe. Verranno esaminate due tipi di supporti: dischi e nastri.

C.1 Sottosistemi a dischi

La virtualizzazione che abbiamo visto essere un'esigenza fondamentale per le architetture server, è diventata anche una caratteristica quasi obbligatoria per i sottosistemi di memorizzazione a dischi. Infatti la virtualizzazione è lo sbocco inevitabile dell'evoluzione dei sistemi a disco, passati da semplici esecutori dei comandi elementari impartiti dai server a sofisticate unità dotate di un proprio sistema operativo: basti pensare che nei sottosistemi storage oggi collegati ai mainframe viene usato un sistema AIX e dei processori P6.

Come esempio della virtualizzazione applicata ai sottosistemi a dischi e dei suoi vantaggi, illustriamo l'implementazione della tecnologia RAID sui dischi.

Redundant Array of Independent Disk (RAID) è un modo di scrivere le informazioni sui dischi in modo che la perdita di uno o più dischi non impedisca l'accesso ai dati. Le figure da 1 a 5 illustrano varie modalita' di implementazione della tecnologia RAID.

Figura 1 RAID 0

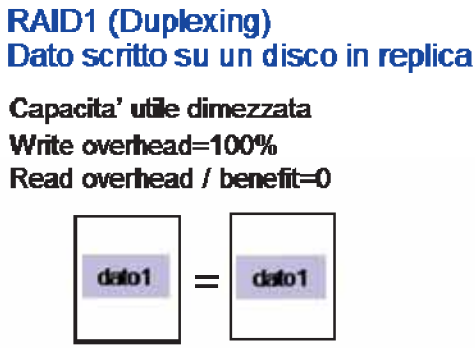


Figura 2 RAID 1

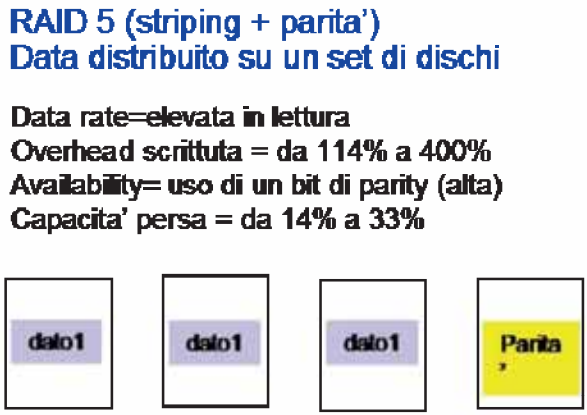


Figura 3 RAID 5

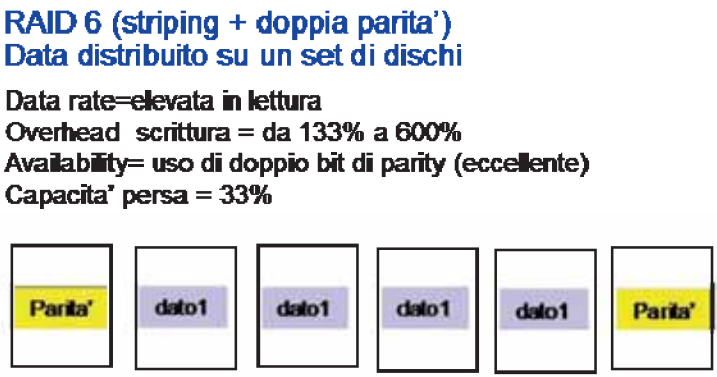


Figura 4 RAID 6

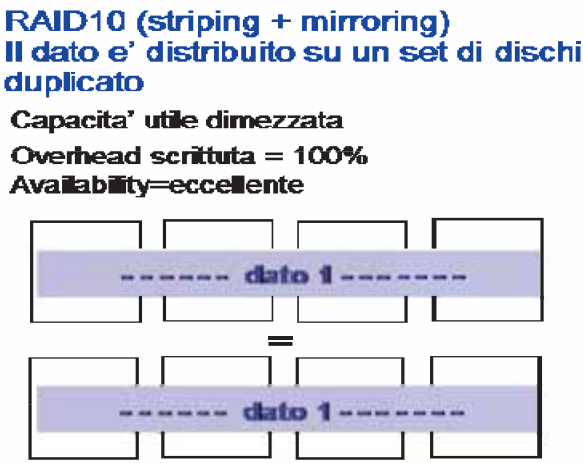


Figura 5 RAID 10

Il concetto alla base della tecnologia RAID è che il singolo blocco di informazioni (singolo dal punto di vista applicativo) viene scritto non su un unico disco ma viene spezzato in parti, e ogni parte scritta su un disco separato. L'aggiunta di un congruo numero di informazioni di parità scritte su altri dischi genera quella ridondanza che permette di accedere al dato anche in caso di perdita di uno o più dischi. Un corollario inscindibile della tecnologia RAID è che maggiore è il livello di protezione che si vuole ottenere, più bit di parità si debbono impiegare e maggiore è la capacità che non può essere usata per i dati (ad esempio, RAID 5 vs RAID 6). Vengono implementate diverse tecniche di RAID, come dalle figure precedenti, per rispondere a requisiti di performance, availability e capacità. La tecnica basata sulla ridondanza delle informazioni risulta più efficente della semplice duplicazione attraverso tecniche di mirroring grazie all'uso di bit di parità. Inoltre il RAID fornisce un data rate elevato grazie alla distribuzione dei dati su più dischi perché consente di avere più meccanismi che lavorano in parallelo su dati striped. Le tecniche maggiormente diffuse sono RAID10, RAID5 e RAID6.

E' chiaro che la tecnologia RAID esige un notevole contributo da parte del SW. In altre parole l'applicazione chiede di scrivere un unico blocco apparentemente indivisibile, ma qualche strato di SW a valle dovrà trasformare questo unico blocco in tanti pezzi (dati + parità) e mandarli in scrittura su dischi separati (scrittura) e inversamente leggere tanti pezzi distinti e aggregarli in un unico blocco per presentarlo all'applicazione (lettura); per non parlare dell'archivio necessario per tenere traccia di questa corrispondenza.

Le prime implementazioni RAID facevano questo lavoro nel Server ma questo risultò subito molto oneroso poichè:

- obbligava a modificare il SW ad ogni nuova funzione di RAID
- consumava potenza pregiata nel server
- Moltiplicava il numero di Operazioni di I/O fra canale e control unit
- La gestione degli errori di I/O risultava particolarmente onerosa.

Quando i sottosistemi storage ebbero potenze di calcolo adeguate, fu possibile implementare il RAID direttamente nel sottosistema storage eliminando gli inconvenienti di cui sopra.

Perciò oggi la virtualizzazione sui dischi permette ai server (mainframe incluso) di pensare ancora in termini di volumi, che ormai sono un'entità puramente logica. Ad esempio lo z/OS crede di avere a che fare con un disco che venne introdotto alla fine degli anni '80 (il 3390).

Il sottosistema storage realizzerà la tecnologia RAID in modo totalmente trasparente. Anche il concetto architetturale di I/O device viene virtualizzato: l'I/O device (e quindi il volume corrispondente) come è visto dai server verrà interpretato dal sottosistema e opportunamente mappato sui dischi reali. La figura 6 illustra la struttura di base di un sottosistema a dischi odierno.

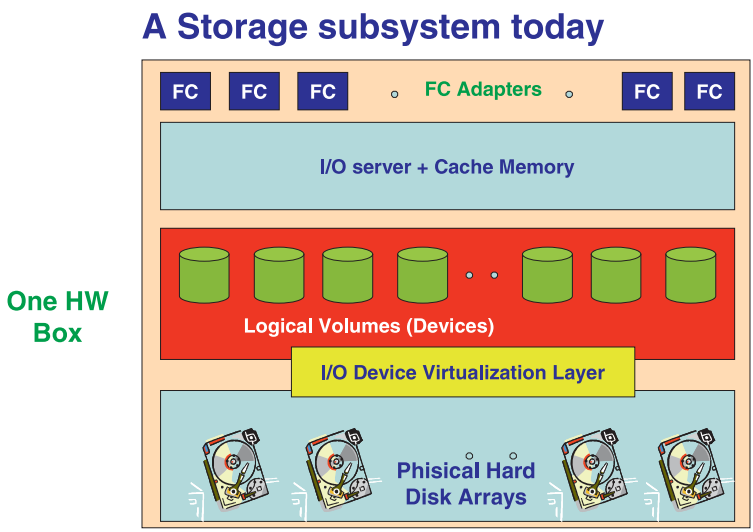


Figura 6 Un sottosistema storage oggi

L'I/O server è dotato di processori molto potenti e di sistemi operativi altamente affidabili (P6 e AIX sui sottosistemi IBM).
Le sue funzioni sono molteplici; fra queste elenchiamo:

- Realizzare la funzione architetturale di Control Unit
- Provvedere al caching dei dati scambiati con i server Questa è una funzione solo in teoria opzionale; in assenza di essa le prestazioni delle applicazioni crollerebbero
- Implementare l'architettura RAID
- Funzioni di Copia dei dati sia all'interno del sottosistema che fra sottosistemi anche geograficamente distanti per realizzare soluzioni di Disaster recovery.

Le componenti di un sistema a disco possono essere divise tra il sistema di Front-End, con cui si intendono gli elementi che interfacciano i Server applicativi, attraverso una rete che consente lo scambio dei dati, e il sistema di Back-End, che indica tutte le componenti che consentono di accedere alla porzione di disco contenente il dato richiesto.

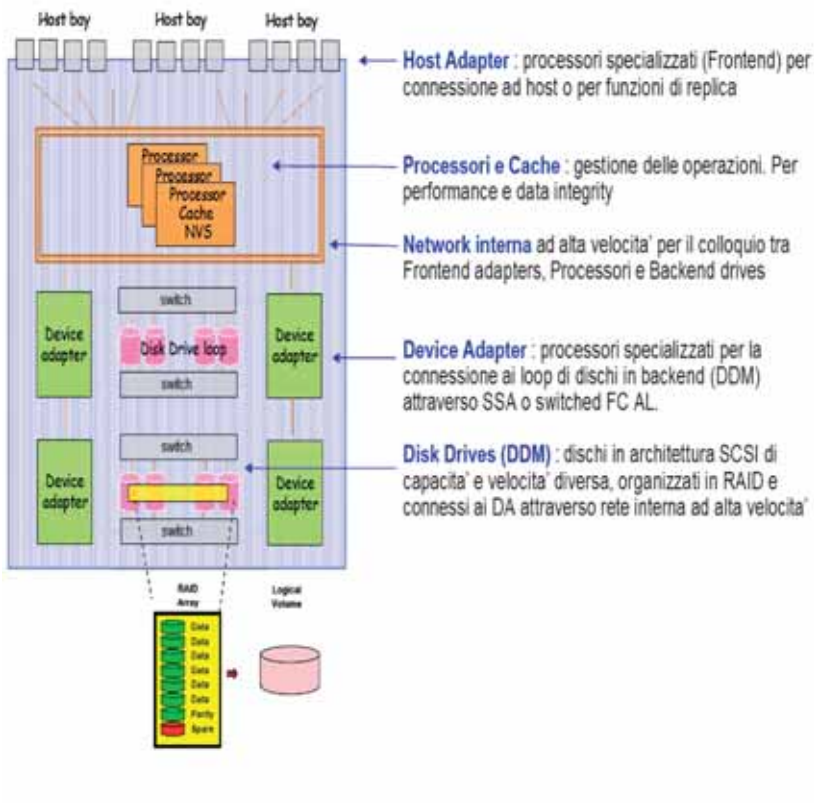


Figura 7 Struttura interna di un sottosistema storage

I volumi logici, così come sono conosciuti dal sistema operativo, vengono allocati distribuendoli su più elementi fisici (Disk Drive, Device Adapter, ...) secondo la modalità di Raid richiesta, oppure per soddisfare lo striping necessario a raggiungere specifiche prestazioni. Le connessioni interne al Sistema Storage tra Front-End e Back-End vengono realizzate in maniera diversa secondo l'architettura scelta, e costituiscono l'elemento qualificante per il raggiungimento dei livelli di servizio (prestazioni, scalabilità, integrità dei dati, ecc.).

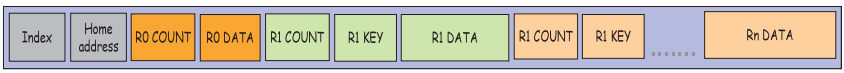
Le caratteristiche principali di un Sistema a Disco sono:

- Supporto per il formato dei dati in ambienti Mainframe (CKD) ed Open (FBA). Il formato CKD è usato per interfacciare i comandi Host ma non è usato come formato dei dati a disco.
- Connettività verso gli Host in Fiber Channel/FICON ed ESCON.

- Virtualizzazione degli SCSI (Small Computing System Interface) device (disk drives) nel Back End in volumi logici raggruppati in Logical Subsystem (LCU). Simulazione della geometria 3390 su device SCSI in architettura RAID.
- Funzioni microcodificate di copia dei dati (locale e remota) e di ottimizzazione delle prestazioni per l'accesso ai dati (es. PAV - Parallel Access Volume).
- Capacità di risolvere la maggior parte delle operazioni a velocità di memoria/network senza coinvolgere i device in backend: tutte le scritture avvengono in cache: 100% write hit.
- Capacità di effettuare le operazioni sui dischi in backend con diverse modalità:
  - Stage : lettura dal device di dati non presenti in cache (cache miss).
  - Destage: scrittura dalla cache a disco di dati modificati.
  - Demote: rimozione di dati non modificati dalla cache.

La virtualizzazione permette di collegare l'idea che i SW hanno della geometria di un disco con la distribuzione fisica delle informazioni sui dischi di back End. Infatti con il formato CKD, ogni traccia sul disco è formattata (dal punto di vista dei sistemi operativi) con una disposizione precisa di record, ciascuno con una funzione ben precisa per l'individuazione del dato:

- 'index' è l'inizio della traccia
- 'home address' è l'indirizzo fisico della traccia
- 'R0' è il record zero che descrive la traccia
- seguono tutti i record di dati in formato Count, Key, Data oppure Count, Data.



Es: RECFM=FB, LRECL=80, BLKSIZE= 4560
ogni Data Area conterra' 4560 bytes, divisi in record logici da 80 bytes
ogni traccia (device type 3390) conterra' 11 record fisici per 50.160 bytes

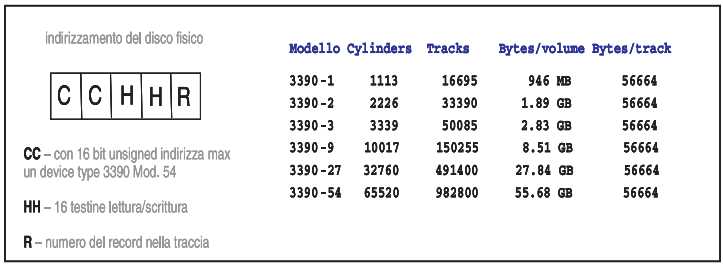


Figura 8 Formato di una traccia in formato CKD

Ad esempio, facendo riferimento allo z/OS, la traccia 0 del cilindro 0 avrà questo contenuto:

1. Index
2. Home Address
3. R0
4. R1, R2 riservati se si vogliono mettere i blocchi che vengono letti all'IPL (e queste posizioni sono definite dall'architettura manframe).
5. R3 è il VOLSER
6. Altri record, da R4 fino alla capienza della traccia, di dati appartenenti a qualche Data Set.

Ma come questa struttura venga poi fisicamente scritta sui dischi di Back End dipenderà da come il sottosistema implementerà la funzione di RAID che si è scelto di usare. Questa struttura fisica sarà totalmente ignota ai sistemi operativi.

C.1.1 Flusso logico di un'operazione di I/O

La figura 6 illustra i componenti software (facciamo riferimento allo z/OS, ma la struttura e il processo sono fondamentalmete uguali per tutti i sistemi operativi Mainframe) e gli elementi di microcode propri dell'architettura Mainframe coinvolti nel processo di richiesta di I/O verso un device disco o nastro. La figura descrive, in ordine temporale, le azioni intraprese dai seguenti componenti:

1. Programma Applicativo
2. Metodo di Accesso al dato
3. I/O Supervisor dello z/OS
4. Channel Subsystem del Mainframe
5. Componenti del Sistema a Disco

Il processo ha inizio all'atto della richiesta di un programma di leggere o scrivere un record e termina quando la traccia che contiene il dato, sul sistema a disco esterno, viene acceduta.

Nello specifico, le funzioni z/OS per l'accesso ai dati, invocate dai programmi applicativi e per brevità identificate con il componente Access Method nella figura 10, sono molteplici e hanno in comune la possibilità di agire come EXCP (Execute Channel Program) driver. Esistono più componenti dello z/OS, come illustrato sempre in figura 10, che con una operazione di EXCP possono invocare lo IOS Driver EXCP/EXCPVR, e quindi invocare l'API STARTIO che passa il controllo all' I/O Supervisor.

Abbiamo introdotto due termini EXCP driver e IOS Driver.

EXCP driver è un SW che esegue l'istruzione SVC "0". Nella convenzione z/OS (per altri sistemi operativi potrebbe essere un altro numero di SVC) la Supervisor Call "0" innesca il processo che porta al lancio di una operazione di I/O. L'input fondamentale al processo di EXCP è il programma di canale che il metodo di accesso a monte ha preparato. Questo programma di canale è in forma virtuale, cioè tutti gli indirizzi al suo interno (ad esempio le aree da cui leggere o in cui scrivere) sono virtuali; non potrebbe essere altrimenti dato che i metodi di accesso (come tutto il resto del SW) opera in memoria virtuale.

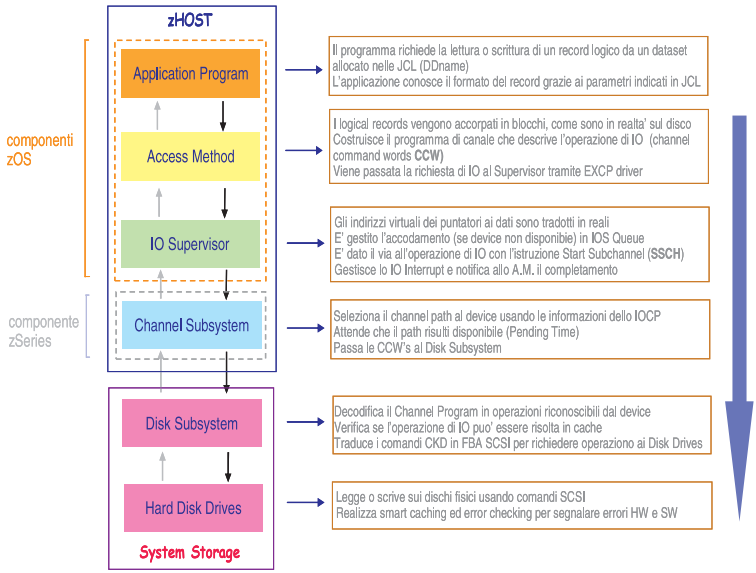


Figura 9 Lancio di un'operazione di I/O

Compito essenziale dell'EXCP è di tradurre il programma di canale da una forma virtuale a una forma reale; all'interno del testo del programma di canale al posto degli indirizzi virtuali vengono messi i corrispondenti indirizzi reali (secondo le tabelle di Dynamic Address Translation).

IOS driver è qualunque SW che interfacci con l'I/O supervisor eseguendo un'API particolare chiamata STARTIO. L'input a questa API è un programma di canale in forma reale. L'IOS provvederà a lanciare l'istruzione Start Subchannel e a reagire al successivo I/O Interrupt.

Nella figura 10 vediamo la relazione fra Application Program, EXPC driver, EXCP, IOS driver e IOS.

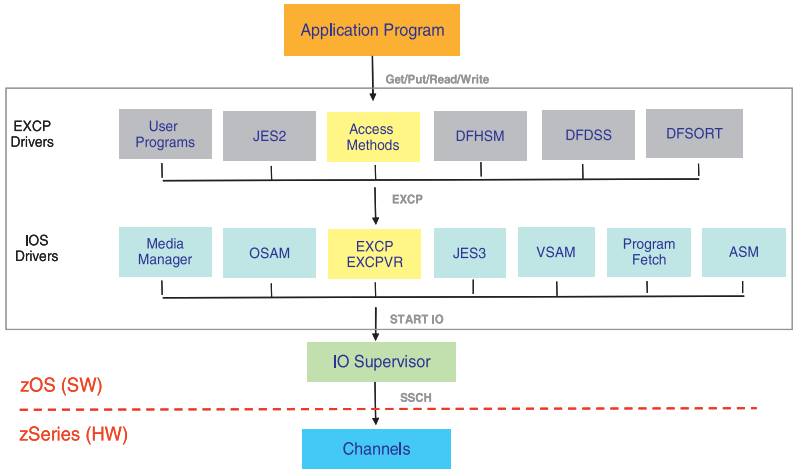


Figura 10 EXCP driver

Quando viene messo in wait il processo che ha chiesto l'operazione di I/O? A valle dell'esecuzione della Start subchannel il controllo ritorna al metodo d'accesso che provvede a eseguire la SVC "1" (Wait) che, sempre nella convenzione z/OS, innesca il processo che sospende un task e lo leva dalle code delle UoW ready.

Vediamo ora invece il flusso logico dell'operazione (figura 11) con cui i dati letti sul disco vengono restituiti al programma che ne ha fatto richiesta (I/O interruption). Il processo visto in precedenza viene ripercorso al contrario coinvolgendo gli stessi elementi del sistema.

Dal sottosistema storage viene presentato al canale un evento di Device End il quale, passato al Channel subsystem, genera un I/O interrupt con tutta una serie di parametri che permettono di collegarlo all'operazione di I/O relativa. L'I/O interrupt handling interfaccia l'IOS, il quale, verificato che l'operazione è terminata correttamente, passa il controllo al metodo d'accesso. Questo estrae dal blocco appena letto il record che l'applicazione ha richiesto e esegue la SVC "2" (Post). Nella convenzione z/OS questa innesca il processo opposto di Wait: la UoW connessa con l'operazione di I/O viene riammessa nella coda delle UoW Ready, potendo nuovamente competere per l'uso dei CP.

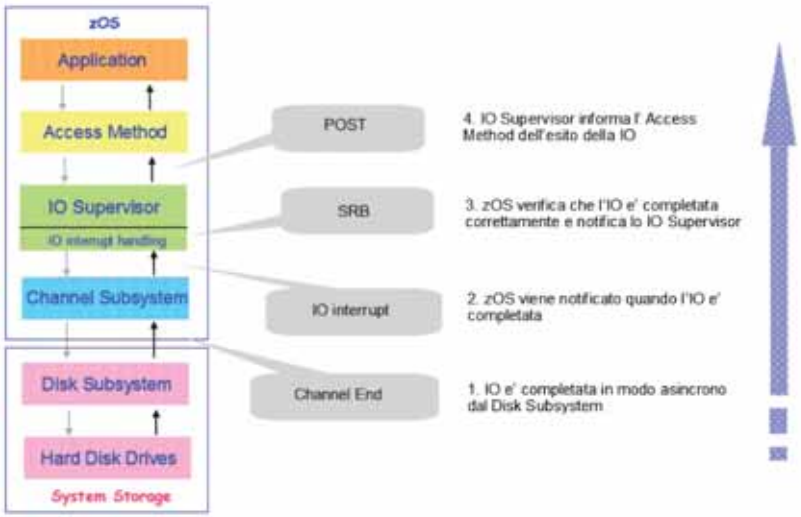


Figura 11 Gestione I/O Interrupt

C.1.2 La gestione della Cache

Nei sistemi a disco, per raggiungere adeguati livelli di prestazione nell'accesso ai dati, è fondamentale il modo in cui l'architettura realizza il concetto di caching. La cache è una memoria elettronica che permette di memorizzare i dati più utilizzati ed accederli in seguito più velocemente. Essa mantiene una lista con i blocchi usati più di recente così da rimuovere i meno utilizzati. La porzione di cache volatile (a rischio perdita dati) contiene solo dati scritti sui dischi in backend.

L'integrità dei dati è preservata garantendone la loro scrittura in una memoria permanente:

1. il dato è scritto in cache
2. una seconda copia è scritta nella parte non volatile della cache – NVS (Non Volatile Storage)
3. è segnalata all' host l'avvenuta scrittura
4. in maniera asincrona il dato verrà scritto sui dischi in backend.

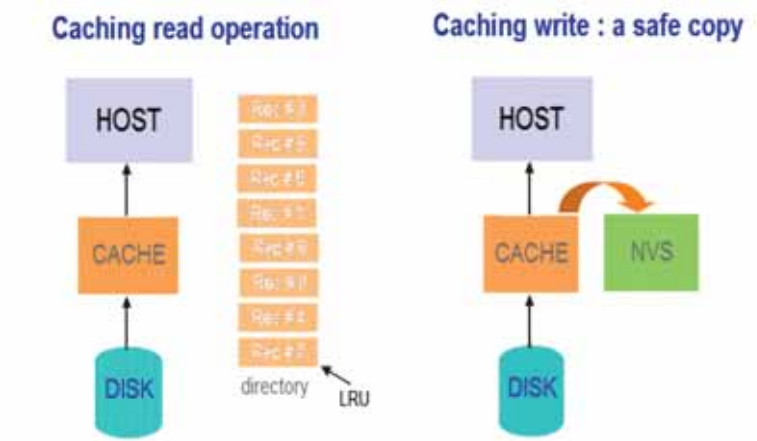


Figura 12 Operazioni in Cache

La tecnologia corrente consente di eseguire le operazioni di read/write cache hits con un tempo nell'ordine di ~ 1ms. Invece, le operazioni da disco fisico sono in genere maggiori di 4ms, poiché richiedono l'intervento di componenti meccaniche, come il movimento delle testine e la rotazione del disco. Ogni disk drive può processare un numero ben preciso di I/O al secondo, determinato da alcuni tempi di operazione:

- Average seek time: tempo di posizionamento della testina sulla traccia richiesta
- Rotational latency: rotazione della superficie del disco finché il primo settore indirizzato passa sotto le testine di read/write heads (avg. time = metà rotazione)
- Transfer time : read/write dei settori di dati (1 sector = 512 Byte).

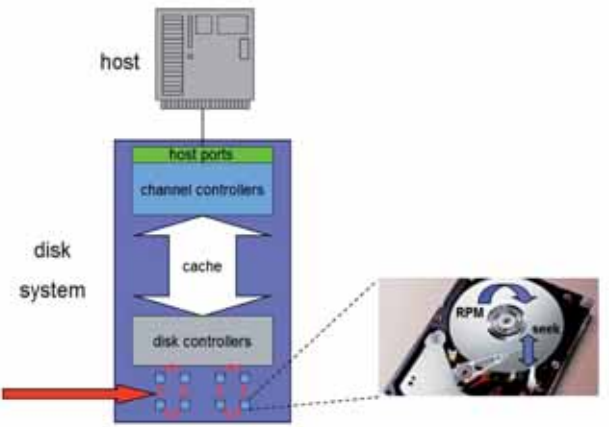


Figura 13 Accesso ai dischi Back End

E' interessante notare che il Transfer Time è limitato da due caratteristiche diverse a seconda che il blocco viene letto dalla cache del sottosistema (cache hit) o dal disco (cache miss):

- per un Cache hit il Transfer Time dipenderà dalla banda passante del canale. Oggi la connessione FC può arrivare a 8Gbit/Sec
- per un Cache miss il Transfer Time dipenderà dalla velocità di rotazione fisica del disco (oggi pari a 15.000 RPM).

C.2 Sistemi Storage a nastro

C.2.0 La tecnologia del nastro

I nastri forniscono grosse capacità di memorizzazione in rapporto allo spazio occupato e hanno, in proporzione, un costo più basso dei dischi. Per tale motivo il nastro è una componente fondamentale nella gerarchia di memorie e su esso si basano le soluzioni di Information Lifecycle Management, che prevedono l'uso dinamico di dispositivi di memoria di caratteristiche diverse sulla base dei requisiti che un dato ha nel suo ciclo di vita.

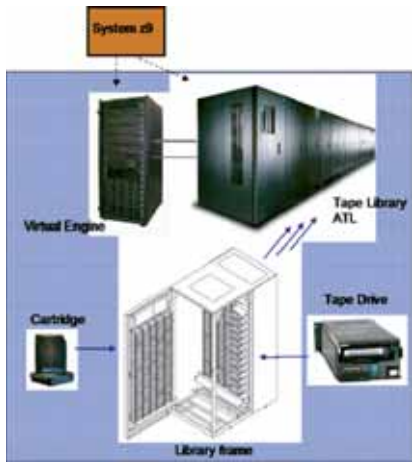


Figura 14 Elementi di una Nastroteca

I nastri normalmente servono a memorizzare i dati inattivi. Sono rimuovibili e portabili (ideali per soluzioni Disaster Recovery) e i dati sulle cartucce possono essere criptati per garantire la riservatezza delle informazioni. Il nastro per sua natura è un dispositivo di tipo sequenziale. Ve ne sono di diversa lunghezza per avere diversi rapporti capacità/prestazioni. Un nastro può contenere diversi Terabyte (TB) di dati, anche grazie alla capacità di compressione dei tape drive; per la memorizzazione di

dati storici di cui debba essere garantita la non modificabilità nel tempo esistono nastri di tipo Write Once Read Many (WORM) che offrono questa caratteristica.

Un nastro memorizza informazioni su piste parallele (es: quasi 1000 tracce vengono scritte in 16 tracce alla volta). Il tape drive usa tecniche di Data Buffer per aumentare le prestazioni nelle letture sequenziali (read-ahead algorithm). Un'operazione di lettura random (mount, ricerca e lettura del dato, unmount) dura poche decine di secondi.

C.2.1 Soluzioni Tape - Componenti

In ambiente Mainframe l' host si connette ad una Control Unit che interpreta il programma di canale e guida le operazioni verso i tape drive. Le operazioni verso i tape possono essere automatizzate attraverso librerie automatiche Automatic Tape Library (ATL) munite di accessori robotici per il montaggio e smontaggio delle cartucce.

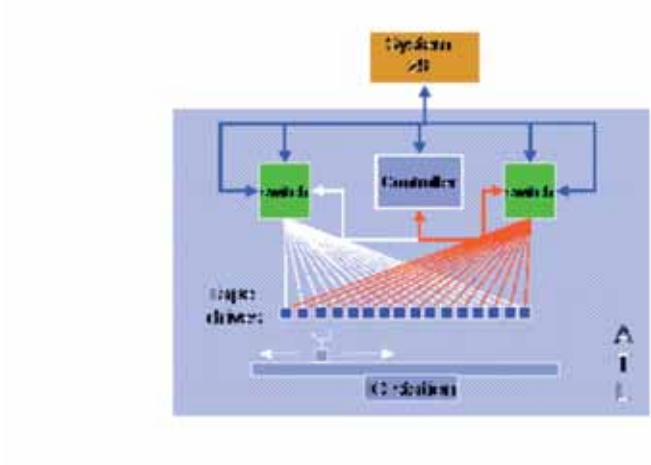


Figura 15 Connessioni Nastro - Mainframe

Il nastro è un device che può essere usato da un solo processo alla volta. In ambienti grandi questo può essere una limitazione. Ha preso piede quindi una nuova forma di virtualizzazione: il Virtual Tape Server (VTS).

Abbiamo un server (l'offerta IBM prevede un sistema AIX) che si interfaccia con il server Mainframe, il quale pensa di avere a disposizione un certo numero di unità Nastro (Virtual drive – sono gli I/O device descritti in IOCDS e IODF). Nel back end questo server AIX in realtà ha a disposizione uno spazio disco (la cui capacità può arrivare a decine di TB) su cui vengono scritti i dati che le applicazioni vogliono scrivere su cartucce. Una struttura di metadati all'interno del VTS provvede a mappare su questo spazio disco i volumi Nastro virtuali (vedi figura 16).

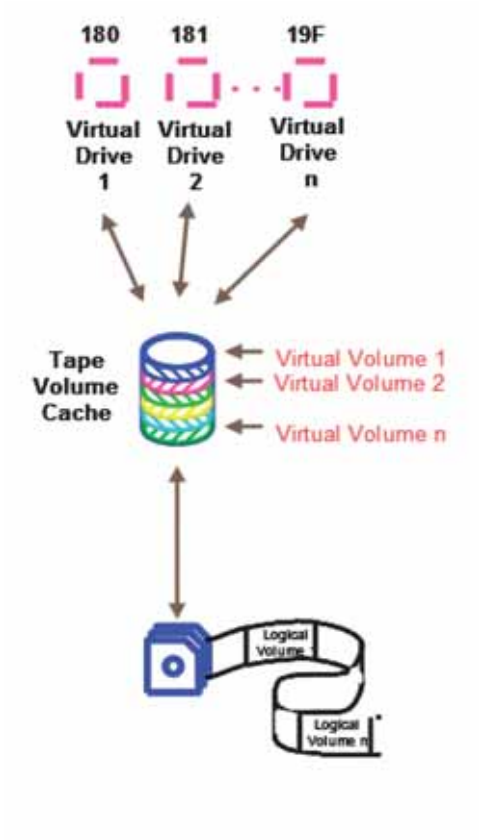


Figura 16 Virtual Tape

Un virtualizzatore di nastri (Tape Virtual Server) fornisce le seguenti funzioni:

- Simula all' host un numero variabile di tape drive virtuali (diverse centinaia)
- Elimina i ritardi indotti dall'accesso al tape fisico poiché i dati vengono scritti sui dischi che costituiscono la cache interna del VTS. I dati vengono scaricati in modo asincrono dalla cache-disco ai nastri utilizzando sistemi di Storage management.
- E' progettato per utilizzare a pieno la capacità delle cartucce e delle librerie di nastri; raggruppa più volumi logici su cartucce fisiche (in backend) permettendo di avere un numero ridotto di tape drives reali.

Il VTS supporta la replica dei dati su una unità remota remota per funzioni di Disaster Recovery.

C.3 Le funzioni di copia dei dati

C.3.0 Le funzioni di replica dei sistemi a disco

Facendo riferimento alle figure 17 e 18, che evidenziano il flusso dei dati tra i volumi sorgenti e quelli target, possiamo identificare tre sistemi di replica che ogni sistema a disco è in grado di implementare, anche se con tecniche diverse:

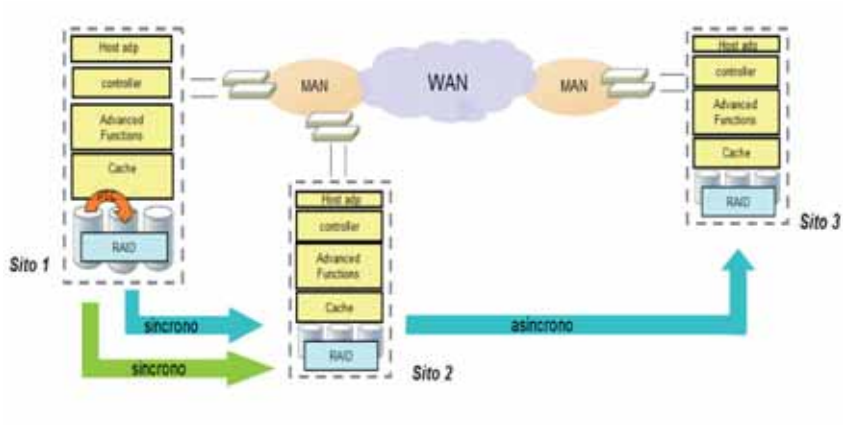


Figura 17 Scenario per la replica dei dati

- Copia Locale: è una copia dei dati istantanea tra volumi all'interno dello stesso sistema Storage.
- Copia Remota tra 2 siti: si distingue in Copia Sincrona, quando la copia dei dati avviene, senza perdita di transazioni, tra sistemi Storage a distanza metropolitana, e Copia Asincrona, se la copia ammette una perdita di dati ed avviene tra sistemi Storage su distanza geografica.
- Copia Remota tra 3 siti: è la copia dei dati sincrona verso il sito più vicino, per garantire la Continuità Operativa, ed asincrona verso il sito più lontano, per finalità di Disaster Recovery.

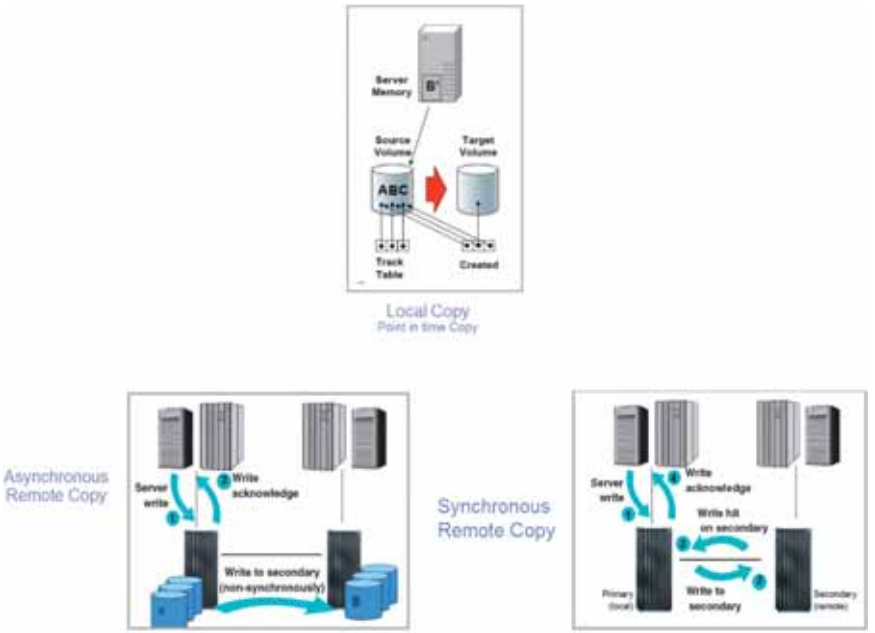


Figura 18 Modalità di replica dei dati nei sistemi a disco

C.4 Connettività e trasporto dei dati

Il protocollo usato sulla rete che connette il mainframe ai dispositivi Storage esterni è il FICON, che ha ormai quasi del tutto sostituito il protocollo ESCON. Esso deriva dallo standard di mercato del protocollo Fiber Channel, usato invece per lo scambio dei dati direttamente tra Sistemi Storage in modalità di replica.

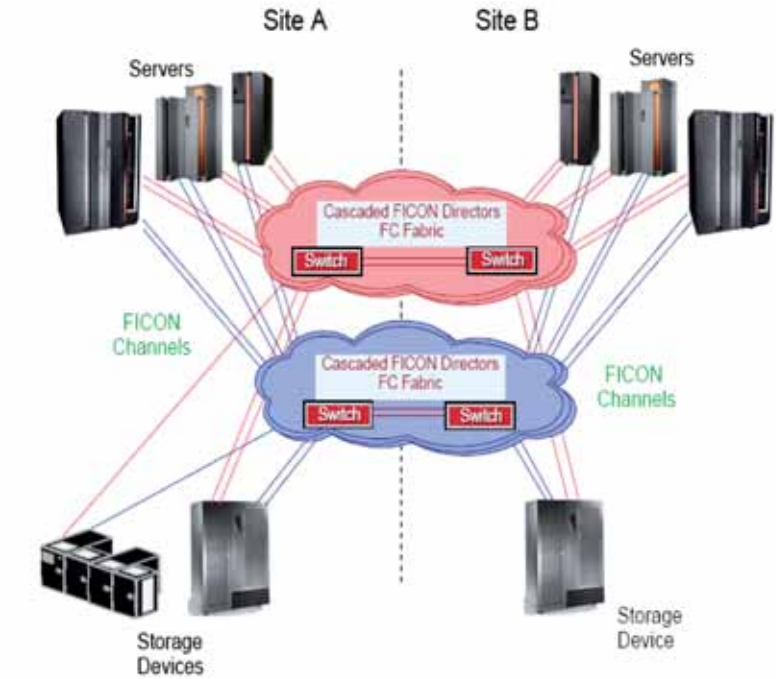


Figura 19 Fibre Channel fabric in ambiente Mainframe

La Fiber Channel Architecture è un insieme di regole che definisce lo standard da implementare per il trasferimento di dati tra computer e device su fibra ottica; FICON è il protocollo che implementa in ambiente Mainframe la Fiber Channel Architecture.

La rete tra server e storage può essere estesa oltre la portata nativa dei canali FICON attraverso l'uso di appositi dispositivi (Channel extender, Dense Wavelength Multiplexing Division (DWDM), ...). Tra questi, i FICON director consentono di connettere ambienti Mainframe a distanza metropolitana, riducendo la complessità della rete. Un canale FICON supporta più comandi di trasferimento (e quindi più operazioni di I/O) simultanei.

Elenco delle figure

1.0 Introduzione ai Sistemi Centrali 12
Figura 1 Sistema centrale 13
Figura 2 Infrastruttura informatica 14
Figura 3 Piattaforma informatica 15
Figura 4 Modello applicativo Host Centrico 16
Figura 5 Modello applicativo Client/Server 17
Figura 6 Modello applicativo Web Application 18
Figura 7 Coreografia SOA 19
Figura 8 Modello applicativo SOA 20
Figura 9 Tipologia di lavoro: Batch job e transazione online 21
Figura 10 Compiti e Ruoli 22
1.1 Architettura dei Sistemi Mainframe 23
Figura 1 Struttura della macchina di von Neumann 23
Figura 2 Schema di collegamento a Bus multiplo 24
Figura 3 Organizzazione di un sistema mainframe secondo la z/Architecture 27
Figura 4 Esempio di virtualizzazione 28
Figura 5 Real e Auxiliary storage per la memoria virtuale 31
Figura 6 Frame, Page e Slot 32
Figura 7 Il processo di traduzione dinamica degli indirizzi 32
Figura 8 Basic Instruction Format 35
Figura 9 Principali istruzioni della z/Architecture 36
Figura 10 Connessione Server e I/O device 39
Figura 11 Un I/O device condiviso fra due server 41
Figura 12 Configurazione di I/O con uno switch FC 42
Figura 13 Esecuzione di un programma di canale (Storage) 44
2.0 Hardware dei Sistemi Centrali IBM (z10EC) 46
Figura 1 Building Blocks z10 EC 46
Figura 2 Chip del processore z10 47
Figura 3 Le unità del core del microprocessore z10 49
Figura 4 Confronto tra le pipeline del processore z10 e quelle dei suoi predecessori 50
Figura 5 Il chip SC-SMP Hub 52
Figura 6 z10 MCM 53
Figura 7 Vista laterale di un book z10 55
Figura 8 Dettagli di un book z10 55
Figura 9 Comunicazione tra i Book 57
Figura 10 Multitiered networks (a) distributed (b) simplified with zIIP+zAAP 59
Figura 11 Il Channel Subsystem z10 60
Figura 12 La definizione IOCP 62
Figura 13 Flusso delle informazioni per le operazioni di I/O 64
Figura 14 Le connessioni book – I/O cage 66
Figura 15 2HMC, SE, Server 67
Figura 16 Interno del sistema z10 68
Figura 17 Confronto tra protocollo FICON e zHPF 69
Figura 18 Confronto tra le performance dei canali FICON, con e senza zHPF 70
Figura 19 Un mainframe inserito in una SAN usando il protocollo FCP 71
2.1 Hardware di IBM zEnterprise 72
Figura 20 Chip dell'elaboratore IBM zEnterprise 196 (z196) 72

Figura 21 Confronto tra le cache dei processori z10 e z196 73
Figura 22 Disegno logico di utilizzo della cache interna 73
Figura 23 MCM (Multiple Chip Module) di zEnterprise 196 74
Figura 24 Confronto tra MCM di zEnterprise 196 e z10 74
Figura 25 La memoria RAIM dello zEnterprise 196 76
2.2 La Virtualizzazione 77
Figura 1 Virtualizzazione....77
Figura 2 Risorse reali e virtuali....78
Figura 3 Condivisione = Risparmio....79
2.3 Virtualizzazione HW e SW 80
Figura 4 Virtualizzazione HW e Partizionamento logico 82
Figura 5 PR/SM Highlights 83
Figura 6 Numero Max LPAR....84
Figura 7 Esempio di definizioni 85
Figura 8 MIF....86
Figura 9 Effetto MP....88
Figura 10 MIPS vs Numero CP 89
Figura 11 Il partizionamento logico riduce l'effetto MP 89
Figura 12 Come le LPAR riducono l'effetto MP 90
3.0 Il Sistema Operativo z/OS.... 91
Figura 1 Dispatching in un sistema multiprocessore 94
Figura 2 Memoria virtuale, reale, ausiliaria 95
Figura 3 Differenti memory addressing....96
Figura 4 Struttura di un Address Space 98
Figura 5 Struttura di un Address Space (dettaglio)....99
Figura 6 Memoria virtuale e Memoria reale....101
Figura 7 Logon al TSO 106
Figura 8 Linea Comandi TSO 107
Figura 9 Interfaccia ISPF....107
Figura 10 Interfaccia ISPF - Menu....108
Figura 11 UNIX System Services 109
Figura 12 Esempio di script UNIX....109
Figura 13 Modi per accedere alla shell UNIX....110
Figura 14 Le differenti interfacce verso z/OS 111
Figura 15 Elaborazione batch con JES....112
Figura 16 SYSIN e SYSOUT 113
Figura 17 Funzionamento di un programma Batch 115
Figura 18 Nomi simbolici e reali in JCL 116
Figura 19 Scelte possibili di SDSF....119
Figura 20 Pannello principale SDSF 120
Figura 21 Schermata SDSF - DA 121
Figura 22 Schermata SDSF....121
Figura 23 SYSOUT dataset del job MIRIAM2....122
Figura 24 VTOC e Data Set....125
Figura 25 Struttura a cataloghi 126
Figura 26 Record e Blocchi all'interno di un Dataset in z/OS 129
Figura 27 Comandi VSAM 130
Figura 28 Struttura di un File System 131
Figura 29 Cilindri e tracce in un disco....131
Figura 30 Il blocco Count Key Data 133
Figura 31 Esempio di Tabella di database relazionale 135

Figura 32 Le componenti del Linguaggio SQL 137
Figura 33 Stored Procedure e Traffico di Rete....138
Figura 34 DB2 for z/OS: un RDBMS Ibrido (Relazionale / XML) 138
Figura 35 DB2 Address Space....140
Figura 36 Esempio di database gerarchico....141
Figura 37 Infrastruttura delle applicazioni....142
Figura 38 Fasi dello sviluppo di un applicazione 143
Figura 39 Language Environment 146
Figura 40 Fasi di compilazione e linkage editor 146
Figura 41 Sottosistemi Transazionali 148
Figura 42 CICS 149
Figura 43 CICS/TM 150
Figura 44 Un sistema IMS 151
Figura 45 Le regioni IMS 151
Figura 46 WAS in z/OS....152
Figura 47 WAS CR & SR 152
Figura 48 - Ambiente di Data warehouse e di applicazioni BI 155
Figura 49 Come alimentare un Data Warehouse 156
Figura 50 Utilizzo processore specializzato zIIP....159
Figura 51 Obiettivi principali della prima release di z/OS MF 160
Figura 52 z/OS MF: Struttura interna 160
Figura 53 Dettaglio del menu della Welcome page dello z/OS MF 161
Figura 54 Struttura hardware dei livelli di memoria in un ambiente multi book 163
Figura 55 Numero di cicli e tempi di accesso alla memoria 163
Figura 56 Gestione verticale nell'Hiperdispatch 165
Figura 57 Parallel Sysplex....168
Figura 58 Coupling Facility 169
Figura 59 Equivalenza delle funzioni di MVS e z/OS UNIX....171
Figura 60 Organizzazione dello z/OS UNIX file system....172
3.1 La Virtualizzazione Software z/VM 173
Figura 1 Condivisione risorse - ambiente z/VM....173
Figura 2 Ambiente z/VM....174
Figura 3 Sistemi operativi ospiti - ambiente z/VM....176
Figura 4 z/VM di secondo livello....177
Figura 5 Esempio di ambiente z/VM 178
Figura 6 Esempio di comandi CP (QUERY VIRTUAL ALL) 179
Figura 7 I Dischi CP_OWNED....182
Figura 8 Esempio di definizione in z/VM User Directory 184
Figura 9 Ambiente z/VM e rete 188
Figura 10 Virtual Network Interface Card (NIC) 189
Figura 11 z/VM Guest LAN....189
Figura 12 z/VM Virtual Switch....190
Figura 13 Configurazione Virtual Switch con VLAN 191
Figura 14 Configurazione Virtual Switch senza VLAN 191
Figura 15 Livelli di Virtualizzazione....192
Figura 16 Virtualizzazione della CPU....193
Figura 17 z/VM virtual memory....194
Figura 18 Liste di Dispatcher 195
Figura 19 Classi di Dispatching 196
Figura 20 Start Interpretive Mode (SIE) 197
3.2 Linux nei Sistemi Centrali IBM....199

Figura 1 Alcuni utilizzi di Linux....200
Figura 2 Modalità operative di Linux su mainframe....202
Figura 3 Modalità di accesso ai device....203
Figura 4 zLinux e processori mainframe....204
Figura 5 Connettività di Linux in z/VM....205
Figura 6 Guest Linux di un sistema z/VM....206
Figura 7 Clonazione di sistemi zLinux....208
3.3 Il Sistema Operativo z/VSE....209
Figura 1 Riepilogo delle principali denominazioni e versioni del VSE dal 1965 ad oggi....209
Figura 2 Esempio di struttura di memoria virtuale in z/VSE 4.2....211
Figura 3 Esempio di collegamenti di un sistema operativo z/VSE con device SCSI su LUN indirizzate tramite WWPN....212
Figura 4 Rappresentazione grafica dei connettori VSE....216
Figura 5 La funzione del DB/2 Client in z/VSE....217
Figura 6 Esempio di utilizzo del connettore z/VSE Navigator per controllare la VTOC....218
Figura 7 z/VSE Health Checker per l'analisi della memoria delle partizioni e della SVA....218
Figura 8 Schema generale delle funzioni TCP/IP in VSE....219
Figura 9 CICS Web Interface applicata ad un comando di sistema....221
Figura 10 Rappresentazione grafica del BSM dello z/VSE....222
Figura 11 Condivisione della Directory LDAP dalle ws e dai CICS-TS del VSE....223
Figura 12 Attuale accesso al CICS-TS dello z/VSE in modalità LDAP....223
Figura 13 Esempio di interconnessioni di sistemi distribuiti con sistema mainframe in ambiente z/VSE e Linux....224
Figura 14 Rappresentazione grafica delle connessioni OSA e HiperSocket....225
Figura 15 L'evoluzione delle più recenti versioni dello z/VSE con il concetto base di PIE (Protezione degli investimenti, Integrazione con altri dispositivi o sistemi, Estensione delle funzionalità)....226
Figura 16 Il laboratorio IBM di sviluppo dello z/VSE a Böblingen (Germania)....227
4.0 La selezione della Piattaforma Informatica e l'Ottimizzazione della Infrastruttura....228
Figura 1 Lo Stack per l'Applicazione Informatica....230
Figura 2 Piattaforma Informatica ed Ambiente Operativo....231
Figura 3 Esempi di Piattaforma Informatica....232
Figura 4 Operating Environment....233
Figura 5 Sistema Virtualizzato....234
Figura 6 Infrastruttura Informatica dei Sistemi - Vista Logica....235
Figura 7 Infrastruttura Informatica - Vista Complessiva....236
Figura 8 Portabilità di una Applicazione Informatica....237
Figura 9 Il Ruolo degli Standard....238
Figura 10 Open Source....240
Figura 11 Re Hosting....245
Figura 12 Il Metodo....246
Figura 13 Infrastruttura composta da un insieme di Piattaforme....247
Figura 14 Condivisione = Risparmio di Risorse....250
Figura 15 Dimensionamento di Sistemi Virtualizzati....256
Figura 16 Intervallo di Campionamento....257
Figura 17 Dipendenza del Massimo dall'intervallo di Campionamento....258
Figura 18 Esempio di calcolo di Sistemi equivalenti....261
Figura 19 Rappresentazione Grafica del TCO per 5 diverse Soluzioni....266
Figura 20 Rappresentazione Grafica del ROI per due Soluzioni....268
Figura 21 Tabella dei Costi per l'applicazione BAO....270
Figura 22 Costi di Acquisizione Applicazione BAO....271

Figura 23 Costi Operativi Applicazione BAO 271
Figura 24 Total Cost of Ownership TCO (3 Anni) Applicazione BAO 272
Figura 25 Spesa Annuale Cumulata - Applicazione BAO 273
Figura 26 Ipotesi di Re-hosting 274
Figura 27 Confronto Costi Re-Hosting 275
Figura 28 TCO Re-Hosting 276
Figura 29 ROI Re-Hosting 276
Figura 30 L'infrastruttura è oggi elemento di complessità 278
Figura 31 Riepilogo Virtualizzazione 281
Figura 32 Tipi di Server Consolidation 283
Figura 33 Virtualizzazione e Consolidamento Logico 284
Figura 34 Infrastruttura Applicazione Sportello 285
Figura 35 Caratteristiche delle Soluzioni a confronto 286
Figura 36 Prospetto Costi Applicazione Sportello 286
Figura 37 Costi di Acquisizione Applicazione Sportello 287
Figura 38 Costi Ricorrenti Applicazione Sportello 287
Figura 39 TCO Applicazione Sportello (3 Anni) 288
Figura 40 ROI Applicazione Sportello 289
5.0 La famiglia dei Mainframe 290
Figura 1 La Famiglia del Mainframe nel 2010 290
Figura 2 S/360 291
Figura 3 S/370 292
Figura 4 Evoluzione del ciclo base dei processori CMOS 294
Appendice A: I sistemi ibridi - una possibile evoluzione dei sistemi mainframe 295
Figura 1 Infrastruttura multi-layer 296
Figura 2 Definizione di Sistema General Purpose 298
Figura 3 Cell BE Processor 300
Figura 4 Sistema Ibrido Integrato 301
Figura 5 Il primo esempio di Sistema Ibrido 303
Figura 6 Principali componenti di IBM zEnterprise System 305
Figura 7 Evoluzione della tecnologia nella piattaforma z 306
Figura 8 Risorse disponibili negli elaboratori System z 306
Figura 9 Schema logico di IBM zEnterprise System 307
Figura 10 Funzioni di IBM zEnterprise Unified Resource Manager 308
Figura 11 La nuova dimensione di potenza dei sistemi ibridi 309
Figura 12 Esempio di sistema federato 310
Figura 13 Esempio di sistema ibrido 311
Appendice B: Sicurezza e Crittografia dei Mainframe 312
Figura 1 IBM Security Architecture 313
Figura 2 Catalogazione dei meccanismi di sicurezza 314
Figura 3 Modello di sicurezza dello z/OS 317
Figura 4 Architettura RACF 319
Figura 5 Esempi modello MAC 322
Figura 6 Architettura dell'ambiente crittografico nel System z 328
Figura 7 ICSF – Architettura 329
Figura 8 ICSF – Gestione chiavi 330
Figura 9 Architettura crittografica su zLinux 331
Appendice C: I sottosistemi Storage 332
Figura 1 RAID 0 332
Figura 2 RAID 1 333
Figura 3 RAID 5 333

Figura 4 RAID 6 334
Figura 5 RAID 10 334
Figura 6 Un sottosistema storage oggi 336
Figura 7 Struttura interna di un sottosistema storage 337
Figura 8 Formato di una traccia in formato CKD 338
Figura 9 Lancio di un'operazione di I/O 340
Figura 10 EXCP driver 341
Figura 11 Gestione I/O Interrupt 342
Figura 12 Operazioni in Cache 343
Figura 13 Accesso ai dischi Back End 343
Figura 14 Elementi di una Nastroteca 344
Figura 15 Connessioni Nastro - Mainframe 345
Figura 16 Virtual Tape 346
Figura 17 Scenario per la replica dei dati 347
Figura 18 Modalità di replica dei dati nei sistemi a disco 348
Figura 19 Fibre Channel fabric in ambiente Mainframe 349

Bibliografia – Pubblicazioni - Link

Documentazione di riferimento della z/Architecture

- z/Architecture Principles of Operation *http://www-01.ibm.com/support/docview.wss?uid=isg26480faec85f44e2385256d5200627dee*

Introduzione al mainframe e ai suoi sistemi operativi

- Introduction to the New Mainframe: z/OS Basics *http://www.redbooks.ibm.com/abstracts/SG246366.html*
- Introduction to the New Mainframe: z/VM Basics *http://www.redbooks.ibm.com/redpieces/abstracts/sg247316.html*
- Introduction to the New Mainframe: z/VSE Basics *http://www.redbooks.ibm.com/abstracts/sg247436.html*
- Linux for IBM System z9 and IBM zSeries *http://www.redbooks.ibm.com/abstracts/sg246694.html*
- Introduction to the New Mainframe: Networking *http://www.redbooks.ibm.com/abstracts/sg246772.html*
- Introduction to the New Mainframe: Large-Scale Commercial Computing *http://www.redbooks.ibm.com/abstracts/sg247175.html*
- Introduction to the New Mainframe: Security *http://www.redbooks.ibm.com/redpieces/abstracts/sg246776.html*
- IBM System z Strengths and Values *http://www.redbooks.ibm.com/abstracts/sg247333.html*

E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. of the ACM, Volume 13, Number 6, June 1970, reperibile alla pagina *http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf*

C.J. Date, "Edgar F. Codd August 23rd, 1923 - April 18th, 2003 A Tribute", 2003, reperibile alla pagina *http://www.sigmod.org/codd-tribute.html*

K. Beyer and al., "DB2 goes hybrid: Integrating native XML and XQuery with relational data and SQL, IBM Systems Journal, Volume 45, Number 2, 2006, reperibile alla pagina *http://www.research.ibm.com/journal/sj/452/beyer.html*

Hardware

- IBM System z10 *http://www-03.ibm.com/systems/z/news/announcement/20080226_annc.html*
- IBM zEnterprise System: Smart Infrastructure for Today's Heterogenous Business Applications http://www.redbooks.ibm.com/abstracts/redp4645.html

- Using IBM System z As the Foundation for Your Information Management Architecture
  http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/redp4606.html?Open
- IBM zEnterprise System Technical Introduction
  http://www.redbooks.ibm.com/redpieces/abstracts/sg247832.html
- IBM zEnterprise System Technical Guide
  http://www.redbooks.ibm.com/redpieces/abstracts/sg247833.html
- IBM System z Connectivity Handbook
  http://www.redbooks.ibm.com/redpieces/abstracts/sg245444.html
- IBM System z BladeCenter Extension Model 001
  http://www.redbooks.ibm.com/redpieces/abstracts/redp4668.html

ABC della Programmazione di sistema dello z/OS

- **Volume 1:** Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
  http://www.redbooks.ibm.com/abstracts/sg246981.html
- **Volume 2:** z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment
  http://www.redbooks.ibm.com/abstracts/sg245652.htm
- **Volume 3:** Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMStvs
  http://www.redbooks.ibm.com/abstracts/sg246983.html
- **Volume 4:** Communication Server, TCP/IP, and VTAM
  http://www.redbooks.ibm.com/abstracts/sg245654.html
- **Volume 5:** Base and Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex (GDPS) *http://www.redbooks.ibm.com/abstracts/sg246985.html*
- **Volume 6:** Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, and Enterprise identity mapping (EIM)
  http://www.redbooks.ibm.com/abstracts/sg246986.html?Open
- **Volume 7:** Printing in a z/OS environment, Infoprint Server and Infoprint Central *http://www.redbooks.ibm.com/abstracts/sg246987.html*
- **Volume 8:** An introduction to z/OS problem diagnosis
  http://www.redbooks.ibm.com/redpieces/abstracts/sg246988.html
- **Volume 9:** z/OS UNIX System Services
  http://www.redbooks.ibm.com/abstracts/sg246989.html
- **Volume 10:** Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and DS8000
  http://www.redbooks.ibm.com/abstracts/sg246990.html
- **Volume 11:** Capacity planning, performance management, WLM, RMF, and SMF
  http://www.redbooks.ibm.com/abstracts/sg246327.html

z/VSE

- **http://www-03.ibm.com/systems/z/os/zvse/documentation/**

Linux

- **http://www.ibm.com/developerworks/linux/linux390/**

Blog

- **http://mainframe.typepad.com**


zEnterprise