SOA Performance on Linux: Process Entry Point

Tuning WebSphere and Linux for Business Processes

Examining the Impact of Server Virtualization and WebSphere Virtual Enterprise on Business Processes in the Linux environment

Khoa Huynh
Vivek Kashyap
Mark Peloquin

IBM Systems & Technology Group
January 2009

Contents

Abstract **3**
1. Introduction **4**
2. Background Information & Performance Evaluation Methodology **4**
2.1. WebSphere Application Server (WAS) Structure 4
2.2. WebSphere Process Server (WPS) Environment 4
2.3. WebSphere Virtual Enterprise (WVE) Environment 5
2.4. Benchmark 6
3. Systems Configurations **8**
3.1. Hardware 8
3.2. Software 9
4. Performance Results **10**
4.1. Tuning Business Process Management (No Clustering) 10
4.2. Tuning Dynamic Operations with WebSphere Virtual Enterprise (WVE) 15
5. Conclusions **26**
References **29**

Abstract

This paper is the third in a series of white papers on Service Oriented Architecture (SOA) performance in the Linux® environment. IBM® has defined five SOA Foundation Entry Points to help a business get started with SOA. This paper focuses on the third entry point – the Process Entry Point. We look at how to tune hardware, Linux, WebSphere®, and DB2® components to achieve optimal performance for hosting business processes implemented in Business Process Execution Language (BPEL). Additionally, we examine the impact of server virtualization and WebSphere Virtual Enterprise's dynamic operations on the performance of business processes in the Linux environment. WebSphere Virtual Enterprise provides a capability to manage a dynamic cluster of application servers that can be activated on a pool of physical servers in response to changes in the workload mix to meet user-defined performance goals. In this study, we include IBM X-Architecture® servers as well as IBM Power™ servers with PowerVM™ virtualization technology in the physical server pool. For performance measurements, we use a benchmark which models the business processes and Web services provided for a typical automobile insurance company.

1. Introduction

IBM has defined five Service Oriented Architecture (SOA) Foundation Entry Points to help businesses get started with SOA in their enterprise environment. These five entry points are *People*, *Process*, *Information*, *Connectivity*, and *Reuse* [1]. This paper—the third in a series of white papers on SOA performance in the Linux environment—focuses on the *Process* entry point, which encompasses the Business Process and Service Choreography aspect of SOA. In particular, this paper takes a close look at performance tuning of business process models implemented in Business Process Execution Language (BPEL) using WebSphere products on Linux. We also look at the performance impact of server virtualization and WebSphere Virtual Enterprise's dynamic operations on business processes.

2. Background Information & Performance Evaluation Methodology

In this paper, we evaluate the performance of business processes implemented in BPEL utilizing the service integration and choreography features of **WebSphere Process Server (WPS) V6.1.** Let us first discuss some basic WebSphere terminology and structure.

WebSphere Process Server is an SCA-compliant runtime element that provides a fully converged, standards-based process engine [3]. The foundation of WPS is the **WebSphere Application Server (WAS)**. WAS is the IBM implementation of the Java® 2 Enterprise Edition (J2EE) platform, which conforms to V1.4 of the J2EE specifications [2].

2.1. WebSphere Application Server (WAS) Structure

A *base* WebSphere Application Server (WAS) installation includes everything needed for a single application server instance. Additional server definitions can be logically grouped into *nodes*. A node can contain many application servers, but cannot span multiple physical servers. A single physical server can have multiple nodes installed on it, each with multiple managed application servers. Multiple nodes can be grouped together into a *node group* or into another logical grouping called a *cluster*. A *cell* contains one or more node groups and/or clusters.

A WebSphere *Network Deployment* installation provides centralized administration and workload management for a cell. A cell has a master administrative repository that holds the configuration information for all nodes in the cell. There is a *Deployment Manager* through which all nodes in the cell can be managed. The Deployment Manager has a GUI called the *Integrated Solutions Console* through which a WebSphere administrator can perform everything from managing a node to deploying an application to any node or cluster in the cell. The Deployment Manager communicates to each node through a *node agent*. The node agent, which is a specialized application server, must be started on a node before the Deployment Manager can "see" it.

WebSphere *Virtual Enterprise* extends the Network Deployment environment, providing on-demand capabilities which allow a dynamic cluster of application servers to respond in real time to changing workload demands, thereby improving the operational efficiency of the servers.

2.2. WebSphere Process Server (WPS) Environment

Every WebSphere Process Server (WPS) environment involves three fundamental layers: WPS applications, a messaging infrastructure, and one or more relational databases. More specifically:

- WPS applications include the process server infrastructure code, such as the Business Process Choreographer (BPC) and any user applications that exploit the process server functions. These applications require a WPS application server to be installed and run.

- A messaging infrastructure is required for WPS and uses four WebSphere Service Integration (SI) buses:
    - Two buses for the Service Component Architecture (SCA) support (SCA.SYSTEM and SCA.APPLICATION buses)
    - One bus for the BPC (BPC SI bus)
    - One bus for Common Event Infrastructure (CEI) asynchronous event publishing (CEI bus)

  When an application server (or a cluster of application servers) is added to a SI bus, a Message Engine (ME) is created. The ME is the component in the WAS process which implements the logic of the messaging infrastructure.

- Relational databases are required for WPS and the messaging infrastructure to store certain application configurations, runtime information, and persistent data. There are two main databases that must handle much traffic in a WPS environment:
    - The Business Process Choreographer database (BPEDB) which stores data objects related to business processes
    - The Service Integration Bus database (SIBDB) which stores data objects for events and message persistence

2.3. WebSphere Virtual Enterprise (WVE) Environment

In our setup, we also use **WebSphere Virtual Enterprise (WVE) V6.1**, which provides application server virtualization, resource management, and a host of advanced operational facilities, such as performance and health monitoring. This combination of capabilities is sometimes collectively referred to as *dynamic operations*. One key feature of dynamic operations is that they can respond in real time to changes in the workload mix (without human intervention if so desired) to ensure that performance goals set by the user can be met. Following are the key elements and features of WVE:

- A *dynamic cluster* of application servers which run the same applications. Applications are first installed and configured on the dynamic cluster. They are then propagated to all active application servers in the cluster.

- A *service policy* that defines a performance goal for one or more applications.

- An *On-Demand Router (ODR)* which is a gateway through which Web service requests flow to the back-end dynamic cluster of application servers. The ODR's primary functions include classification of incoming requests (based on rules defined by the user) and intelligent request routing based on sense and response mechanisms from the back-end servers.

- A *Web server* which is placed in front of the ODR as a trusted proxy server. In this study, we choose to use the *IBM HTTP Server*.

- Web service requests will flow through the Web server and be intelligently routed to active application servers in the dynamic cluster by the ODR based on the performance information collected from the cluster members. Workload will be balanced across all servers in the cluster in order to achieve the performance goals specified in the service policy. This dynamic workload balancing is based on load distribution, service policy, and available resources. This capability is provided by three autonomic managers associated with the ODR: the Application Placement Controller (APC), Dynamic Workload Manager (DWLM), and Autonomic Request Flow Manager (ARFM). They make decisions on health management, traffic shaping, and application placement for the ODR.

Dynamic clustering in the WVE environment involves the clustering of WPS applications and the messaging infrastructure. Conceptually, the clustering of WPS applications is not very different from

clustering plain J2EE applications in the WebSphere Application Server environment. WebSphere clustering techniques in the static environment are also applicable to the dynamic WVE environment. These techniques are discussed in [4]. However, clustering the messaging infrastructure is significantly more complex.

If a cluster of application servers is added to a Service Integration (SI) bus, each server in the cluster is capable of running the Message Engine (ME) created for that cluster. However, only one server can have an active instance of the Message Engine at any given time. There are two approaches that a Message Engine can be configured to work with WPS applications:

- The Message Engine is "local" to the cluster of WPS applications. In this case, the ME runs within the same application cluster as the WPS applications.
- The Message Engine is located in its own cluster, separate from the WPS applications. This is also called the "Silver" topology in a classification of WebSphere clustering topologies [4].

In this study, we consider both approaches.

2.4. Benchmark

In this study, we use a benchmark that models the business processes and Web services provided for a typical automobile insurance company [10]. The benchmark specifies a macro workload whose driver can generate an end-to-end workload similar to that of an actual production system in an SOA environment. *Figure 2.1* shows the components of the benchmark.

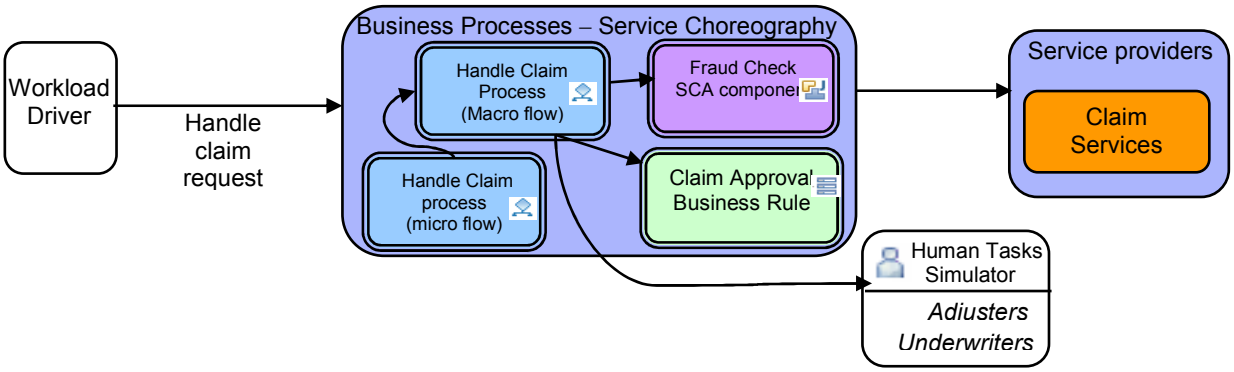

Figure 2.1 – Architecture of benchmark used in our evaluation

The benchmark, as considered in this paper, makes extensive use of IBM SOA platform products in the following areas:

- Enablement of Web services, using IBM WebSphere Application Server (WAS)
- Business process choreography, using integration and choreography features of IBM WebSphere Process Server (WPS)

In the benchmark, the Web services are implemented as part of a *ClaimServices* application. These Web services represent typical services that are involved in the processing of an automobile insurance claim, such as creating a claim, updating a claim, approving or denying a claim, checking insurance coverage, generating a list of approved repair shops, selecting a repair shop, and informing the customer. Some business logic is embedded in the implementation of these services. However, the presence of business logic might hinder us in evaluating the performance of the underlying middleware layers supporting Web services as well as investigating potential problems that might occur. As a result, we decided to run the benchmark in *Infrastructure Mode* which keeps the business logic in the Web services to a minimum; it only performs minimal calculations and returns responses.

The processing of auto insurance claims is done through a BPEL application called *HandleClaim*. *Figure 2.2* illustrates the *HandleClaim* application, which is implemented using the business process choreography features provided by WebSphere Process Server (WPS). After a customer submits a claim, the *HandleClaim* process calls an SCA component called *FraudCheck* to determine whether the claim is fraudulent or not. If a claim is fraudulent, it is rejected, and the processing of the claim is completed. On the other hand, if the claim is found to be valid, an *Adjuster* is called upon to evaluate and update the claim. Next there is a business rule which determines whether the value of the claim is less than $500. If so, the claim is approved automatically; otherwise, it would have to be approved in person by an *Underwriter*. The actions of *Adjusters* and *Underwriters* are simulated by another application – the *HumanTaskSimulator* – which polls and performs the roles of adjusters and underwriters. After much experimentation, we found that setting the *HumanTaskSimulator*'s parameters to the following values was more than adequate for the workloads considered in this study:

- Number of Adjusters = 30
- Number of Underwriters = 20
- Polling period for Adjusters = 200 ms
- Polling period for Underwriters = 400 ms

After a claim is approved, a check is sent to the customer, and the processing of the claim is completed. In this paper, both micro and macro workflows are considered. The underlying business logic of the claim services, such as creating, updating, rejecting, approving, and completing a claim, is performed by calling the Web services implemented in the *ClaimServices* application. In *Figure 2.2*, the claim services are represented by the boxes in blue.

The benchmark's *Workload Driver* is a stand-alone, multi-threaded HTTP client which can generate concurrent insurance claim requests to the *HandleClaim* application using the Service Oriented Access Protocol (SOAP) implemented on top of the HTTP transport protocol [9].

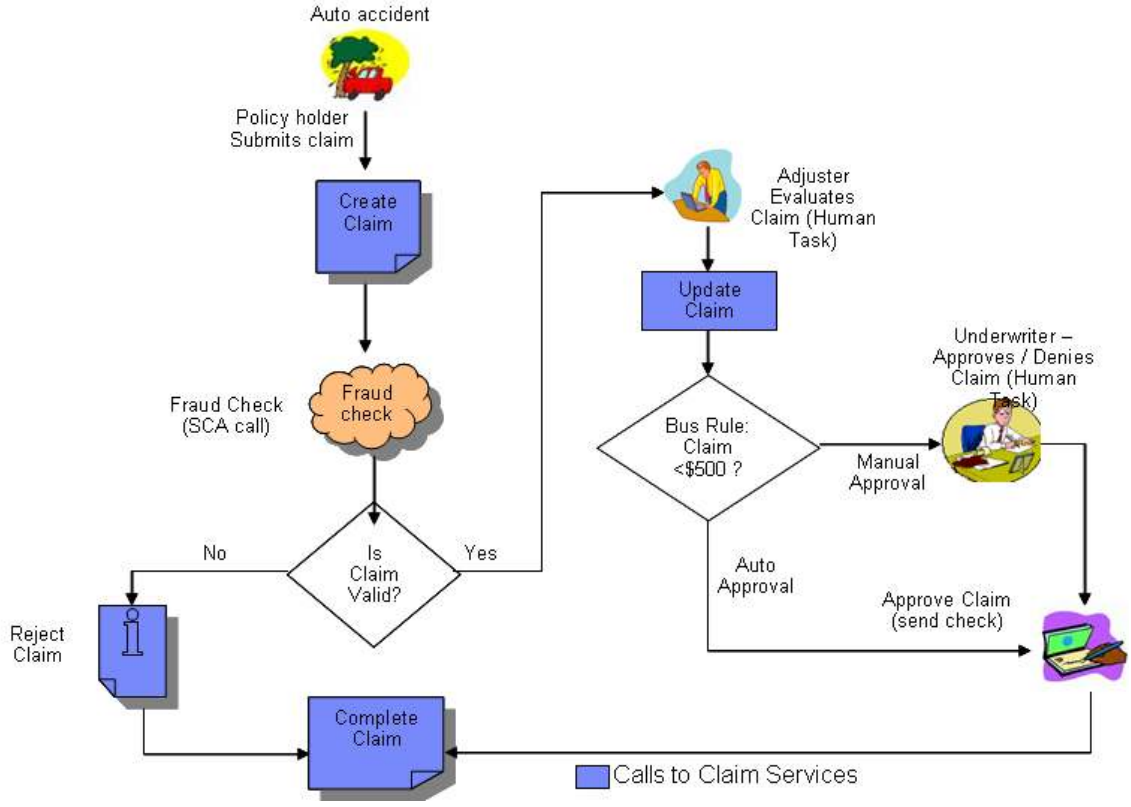

Figure 2.2 – Insurance claim processing workflows

In this paper, we consider both IBM X-Architecture and IBM Power Architecture server platforms.

In most of our tests, we start warm-up runs (which can take as long as 300 seconds depending on the workload level) prior to actual data collection to ensure optimal and consistent results. Warm-up runs were especially needed because, by default, the IBM Java Virtual Machine (JVM) in WAS uses a higher optimization level for compiles, thus resulting in faster runtime performance, but at the expense of slower server startups.

3. Systems Configurations

3.1. Hardware

As mentioned previously, we consider both IBM X-Architecture and Power Architecture server platforms for hosting the benchmark's Web services and business processes.

3.1.1. IBM X-Architecture Servers

In this study, we employ the IBM System x3850 M2 (*Table 3.1*), which implements the IBM eX4 chipset [7,8], to host WebSphere, DB2, and the benchmark's Web services and business process components.

Server	IBM System x3850 M2
CPU	4 x 64-bit Quad-Core Intel® Xeon® Processor X7350 (2.93 GHz)
Memory	64 GB (667 MHz DDR2)
Network	Integrated Dual-Port Gigabit Ethernet w/ TCP-IP off-load engine

Table 3.1 – IBM System x3850 M2 Configuration

The benchmark's Workload Driver runs on an IBM System x3650 (*Table 3.2*).

Workload Driver	IBM System x3650
CPU	2 x 64-bit Quad-Core Intel® Xeon® X5460 (3.16 GHz)
Memory	24 GB (667 MHz DDR2)
Network	Integrated Dual-port Gigabit Ethernet

Table 3.2 – IBM System x3650 Configuration

3.1.2. IBM Power Architecture Servers

For the Power platform, we use an IBM Power 570 (*Table 3.3*) [5] with POWER6 processors [6] to host WebSphere and the benchmark's applications in some tests and two IBM OpenPower 720 servers (*Table 3.4*) [11] with POWER5 processors in other tests.

Server	IBM Power 570
CPU	2 x 64-bit Dual-Core IBM® POWER6® (4.7 GHz), 4 MB L2 cache per core, 32 MB L3 cache shared per two cores
Memory	32 GB (667 MHz DDR2)
Network	Dual-Port Gigabit Ethernet
Internal Storage	1 x SAS controller with 2 x 300 GB, 15K rpm SAS drives
Threading	Simultaneous Multi-Threading (SMT)™ Technology

Table 3.3 – IBM Power 570 Configuration

Server	IBM OpenPower 720
CPU	2 x 64-bit Dual-Core IBM® POWER5® (1.5 GHz), 1.9 MB L2 cache, 36 MB L3 cache
Memory	16 GB
Network	Dual-Port Gigabit Ethernet
Internal Storage	1 x SAS controller with 4 x 36 GB / 72 GB 15K rpm SAS drives
Threading	Simultaneous Multi-Threading (SMT)™ Technology

Table 3.4 – IBM OpenPower 720 Configuration

All servers were connected to a Cisco Systems® Catalyst® 3750 Series Gigabit Switch (Model WS-C3750G-24TS-S).

3.2. Software

The Linux operating system on the IBM System x3850 M2 server is Novell SUSE Linux Enterprise Server (SLES) 10 Service Pack (SP) 1 for AMD64 & EM64T (x86_64).

The Linux operating system on the IBM Power 570 and OpenPower 720 is Novell SUSE Linux Enterprise Server (SLES) 10 Service Pack (SP) 1 for PPC (ppc64).

The operating system on the workload driver system (IBM System x3650) is Novell SUSE Linux Enterprise Server (SLES) 10 Service Pack (SP) 2 for AMD64 & EM64T (x86_64).

All servers run with WebSphere Application Server (WAS) V6.1.0.17, WebSphere Process Server (WPS) V6.1.0.2, and WebSphere Virtual Enterprise V6.1.0.4. All WebSphere products used in this study are 64-bit.

For databases, we use the 64-bit DB2 Enterprise Server Edition V9.1 for Linux.

4. Performance Results

First let us look at how we can create an optimal, non-clustered hardware environment for the Business Process Management (BPM) components, such as WebSphere Process Server, DB2, etc., and how we can tune those components to get the best throughput for our benchmark.

4.1. Tuning Business Process Management (No Clustering)

First, let us consider the setup as illustrated in *Figure 4.1* where everything runs on a single IBM Power 570 server. In this setup, both WPS and DB2 are installed on an IBM Power 570 with dual POWER6 processors (a total of 4 *Processing Units* or cores). Both *HandleClaim* and *ClaimServices* applications are also installed and executed on this server. There is a single disk where the BPEDB, SIBDB, and all WPS logs for transaction and compensation services reside. The data for this configuration is shown in the first two rows in *Table 4.1* – under *Configurations 1* and *1.1*.

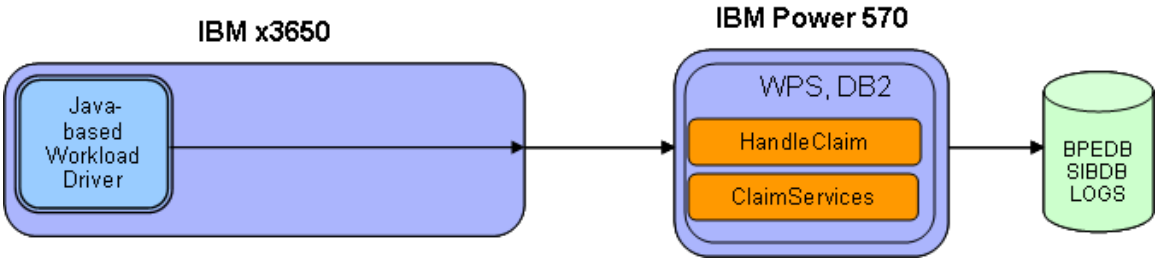

Figure 4.1.1 – Single server setup (Configurations 1 & 1.1)

Config	Max Tx	BTPS	CPU Utilization			DB2 Databases				WPS Logs		
			Single Server	WPS	DB2	Single Disk	BPEDB	BPE Log	SIBDB	Single Disk	Transaction	Compensation
1	80	Error	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1.1	80	2.59	53%	N/A	N/A	99.60%	N/A	N/A	N/A	N/A	N/A	N/A
2	80	21.7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2.1	80	45.7	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2.2	80	46.28	N/A	97%	N/A	N/A	N/A	N/A	N/A	N/A	67%	60%
3	80	61.12	N/A	93%	30%	N/A	30%	13%	12%	N/A	79%	75%
3.1	80	55.84	N/A	89%	26%	N/A	47%	11%	13%	N/A	81%	75%
3.2	80	44.56	N/A	78%	30%	N/A	43%	10%	9%	N/A	70%	60%
3.3	80	41.79	N/A	75%	30%	N/A	70%	N/A	73%	N/A	63%	66%

Table 4.1.1 – Tuning Business Process Management (non-cluster)

As you can see in *Table 4.1.1*:

- **Default Configuration**: ***Configuration 1*** shows the out-of-the-box configuration where there is no tuning whatsoever.

  Result: In this configuration, the benchmark cannot even run. The default configuration for the WAS JVM is too small to handle the necessary processing for the benchmark used in our study.

- **WAS JVM Heap Tuning**: ***Configuration 1.1*** shows what happens when we tune the WAS JVM heap configuration. Two basic JVM heap parameters are the size of the heap and the garbage collection policy. There are several tools available, such as the Tivoli® Performance Viewer (included with WebSphere), that can help analyze and monitor the heap usage and garbage collection so that the heap can be specifically tuned for a particular workload. In our tests, by looking at the number of memory pages allocated to the heap that are actually in use while the

benchmark scenarios are being run, we found that the heap usage is mostly at 2048 MB. As a result, we set the JVM heap size for WAS at 2048 MB. Since the Power 570 server has 32 GB of physical memory, which is more than adequate for our tests, we decided to set the minimum and maximum heap size to the same value (2048 MB) to avoid the overhead of frequent heap size changes. In addition to the JVM heap size, the garbage collection policy can also affect performance. The WAS JVM supports four different garbage collection policies. The default garbage collection policy is **`optthruput`**. However, we decided to use the **`gencon`** (Generation Concurrent) garbage collection policy, which handles short-lived objects differently from long-lived objects, because our earlier testing for Web service scenarios showed that the **`gencon`** policy was more suitable for Web service scenarios [12]. Under the **`gencon`** policy, the heap is split into new and old segments. Long-lived objects are promoted to the old space while short-lived objects are garbage collected quickly in the new space (called a *nursery*). We also set the size of the *nursery* to 1536 MB (75% of the total heap size) through the *Integrated Solution Console (ISC):*

- Go to Servers → Application Servers → *server name* → Server Infrastructure → Java and Process Management → Process Definition → Additional Properties → Java Virtual Machine
- Set the "Initial Heap Size" and "Maximum Heap Size" boxes to 2048 MB
- Enter **`- Xgcpolicy:gencon -Xmn1536M`** in the "Generic JVM arguments" box

Result: Tuning the WAS JVM heap configuration allows the benchmark to run, but its overall throughput (*Business Transactions Per Second* or BTPS) is only 2.59. The data in *Table 4.1.1* for *Configuration 1.1* shows that the CPU utilization for the IBM Power 570 server is 53% while the disk utilization for the single disk is 99.6%. This single disk is obviously the system performance bottleneck.

- **Adding Another Server for DB2: *Configuration 2*** in *Table 4.1.1* shows the result of our effort to remove the single disk as the system performance bottleneck. First, we use another server – the IBM x3850 M2 – as the DB2 server and move the BPEDB and SIBDB databases over to the x3850 M2's disk arrays. This is illustrated in *Figure 4.1.2*. More specifically, we partition the workload as follows:

  - The x3850 M2 server runs the *HumanTaskSimulator*, DB2, and hosts the databases on its disk arrays:
    - BPEDB on a 24-disk array attached to the IBM System Storage DS3400 controller
    - BPE Logs on a 12-disk array attached to the IBM ServeRAID MR10M controller
    - SIBDB on a 12-disk array attached to the same IBM ServeRAID MR10M controller

  - The Power 570 server now only runs WPS and the benchmark's *HandleClaim* and *ClaimServices* applications. We also make some additional improvements on this server's configuration:
    - One disk for transaction logs
    - Another separate disk for compensation (recovery) logs
    - The WAS JVM heap size is increased from 2048 MB to 4096 MB with the nursery size set to 3072 MB (75% of the heap size)

Result: With these changes, the overall benchmark throughput is now 21.7 BTPS – a significant improvement over 2.59 BTPS for *Configuration 1.1*.

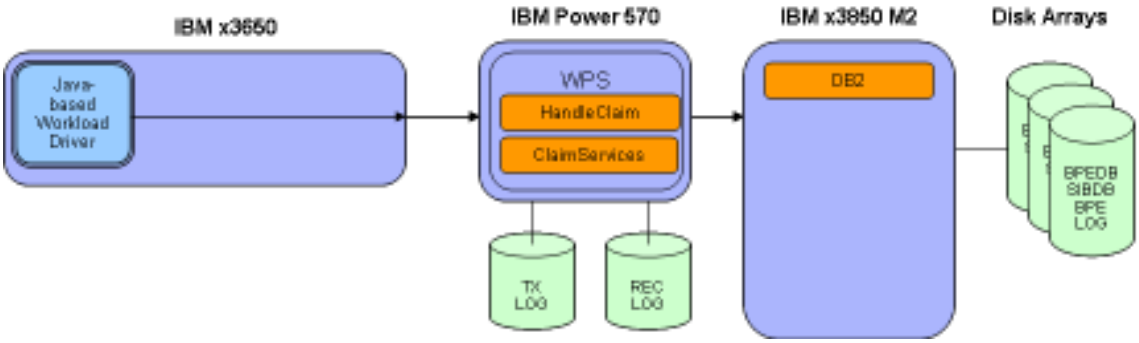

Figure 4.1.2 – Separate WPS & DB2 servers (Configurations 2, 2.1, 2.2, and 3)

- **WAS/WPS Concurrency Optimization**: ***Configuration 2.1*** in *Table 4.1.1* shows the performance impact of additional WAS and WPS tunings that maximize concurrency. These are recommended by the WebSphere Performance Teams [13]. In particular, since the Power 570 server has a total of 4 cores or 8 logical CPUs with Simultaneous Multi-Threading (SMT), we follow the recommendations for 8 CPUs:
  - Thread pools:
    - Default thread pool max = 200
    - Web container's thread pool max = 100
  - Connection pools:
    - BPEDB data source's connection pool max = 150
    - CEI ME data source's connection pool max = 80
    - SIBDB System data source's connection pool max = 30
    - SIBDB BPC data source's connection pool max = 50
  - J2C connection factories:
    - BPECF connection pool max = 40
    - BPECFC connection pool max = 40
    - HTMCF connection pool max = 20
  - J2C activation specifications:
    - BPEInternalActivationSpec → Custom properties → maxConcurrency = 60
    - <Benchmark> → Custom properties → maxConcurrency = 40

  Result: With the concurrency optimizations, we essentially double the benchmark's overall throughput to 45.7 BTPS (as compared to 21.7 BTPS for *Configuration 2*).

- **DB2 Tuning**: ***Configuration 2.2*** in *Table 4.1.1* shows the performance impact of applying DB2 tunings to optimize the log data rates. We set the following DB2 parameters:
  - Max storage for lock list (4KB) = 400
  - Log buffer size (4KB) = 512
  - Log file size (4KB) = 8000
  - Number of primary log files = 10
  - Number of secondary log files = 10
  - Maximum number of active applications = 250
  - Average number of active applications = 50
  - Percent log file reclaimed before soft checkpoint = 300
  - Percentage of lock lists per application = 20
  - Buffer pool size (pages) = 64000

  Result: As can be seen in *Table 4.1.1*, these DB2 tunings yield a benchmark throughput rate of 46.28 BTPS – only a small improvement (1.3%) over what we are able to get for *Configuration 2.1*. *Table 4.1.1* also shows that this *Configuration 2.2* has the CPU utilization of the WPS server (Power 570) at 97% and the disk utilization at 67% and 60%, respectively, for transaction and

compensation (recovery) logs. This indicates that the system performance bottleneck is no longer the disk subsystem, but rather, the Power 570's two dual-core POWER6 processors. Adding more processors to the WPS Server would certainly increase the benchmark throughput – provided that some J2C Connection Factories and Activation Specifications would also be increased to accommodate the higher number of processors as recommended by the WebSphere Performance Team [13].

- **Cache mirroring in the IBM System Storage DS3400**: ***Configurations 3*** and ***3.1*** in *Table 4.1.1* show the performance impact of the cache mirroring capability in the DS3400. Cache mirroring, which is the default setting on the DS3400, can be left enabled for reliability / availability reasons. In our setup, the BPEDB disk array is attached to the DS3400 controller. In *Configuration 3*, we disable cache mirroring to achieve the best benchmark throughput, but in *Configuration 3.1*, we leave the cache mirroring enabled (the default setting).

  Result: As we can see in *Table 4.1.1*, with cache mirroring enabled, the disk array utilization for BPEDB jumps from 30% to 47%, and the benchmark throughput drops from 61.12 BTPS to 55.84 BTPS – an 8.6% performance drop. (Starting with *Configuration 3*, we use a more powerful Power 570 server with 4 dual-core POWER6 processors). This is a classic trade-off between performance and reliability / availability.

- **Reducing WAS JVM Heap Size in Memory-Constrained Environment**: ***Configuration 3.2*** in *Table 4.1.1* shows the impact of reducing the WAS JVM heap size from 4096 MB to just 2048 MB. This could be necessary in memory-constrained environments.

  Result: Reducing the WAS JVM heap from 4096 MB to 2048 MB drops the benchmark's overall throughput from 55.84 BTPS to 44.56 BTPS – a significant 20% performance drop. As a result, it is highly recommended that the WAS JVM heap be adjusted accordingly after adding CPUs to the WPS server.

- **Write-Back Caching in IBM ServeRAID MR10M controller**: ***Configuration 3.3*** in *Table 4.1*.1 shows what happens when we use write-through caching policy in the MR10M disk array controller (instead of write-back). Write-back caching is always good for the performance of disk writes since the disk write operations are considered done as soon as the data is written to the cache (instead of physical disks). Data in cache would then be written ("flushed") to physical disks at a later time. However, in an environment where there is no battery backup for the controller and/or the disk subsystem, write-through caching could help protect the integrity of data when there is a power outage. In our setup, the SIBDB and BPE Log disk arrays are attached to the MR10M controller.

  Result: Using write-though caching policy for the SIBDB disk array results in a 6% performance drop – from 44.56 BTPS with write-back caching in *Configure 3.2* to 41.79 BTPS with write-through caching in this *Configuration 3.3*. Note that, for *Configuration 3.3* in *Table 4.1.1*, the utilization of the disk array for BPEDB jumps to 70% because we now use a single 24-disk aray for both BPEDB and BPE Logs. However, our test data (not shown in *Table 4.1.1*) indicates that this does not have any noticeable impact on the benchmark throughput. This is because, even at 70% utilization, the disk I/O queue depth for the 24-disk array is still less than 1, indicating that the disk array is not the performance bottleneck. In fact, this is lower than the CPU utilization (75%) for the WPS server.

- **Linux Logical Disk Striping**: in *Configurations 3* and *3.1*, the utilization for the disks holding the transaction log and compensation (recovery) log on the WPS server is quite high – around 80% for transaction log and 75% for compensation log. While this is still lower than the CPU utilization, it would be interesting to see if somehow we could lower the disk utilization and see if that would have any impact on the benchmark's overall throughput. We don't have any extra disk array controller that could be installed in the WPS server, but we are able to find two extra disks that could be used to set up a 4-disk logical array through the Linux Logical Volume Manager (LVM) to handle both transaction and compensation log traffic.

  Result: Table *4.1.2* shows the results. For a 4-disk logical array, the stripe size of 64 KB provides better performance than smaller stripe sizes of 4 KB and 16 KB, given the size of each log data write is between 4 KB and 5 KB. The utilization of the device manager is found to be over 90% while the utilization of each individual disk ranges from 27% to 36%. However, dedicating a single disk to each log, without any logical disk striping, produces the best benchmark throughput – even though we use only 2 disks and each disk is utilized more than 75% of the time. This indicates that the overhead of the Linux Logical Volume Manager in managing a logical disk array has a negative performance impact when it comes to transaction and compensation logs on the WPS server. As a result, for the rest of our study, we dedicate a single physical disk for each WPS log.

Disk Setup	Stripe Size (KB)	Benchmark Throughput (BTPS)
Two Single Disks	N/A	55.84
4-Disk Logical Array	4	44.79
	16	49.79
	64	51.38

Table 4.1.2 – Logical Disk Striping (Linux Logical Volume Manager)

- **Linux Large Page Support**: We configure large pages (16-MB page size) on the Power 570 server (WPS server) for the WAS JVM heap and find that it does not result in any performance improvement. In fact, with large pages support, the benchmark's overall throughput goes down from 61.12 BTPS (*Configuration 3*) to 59.70 BTPS – a performance degradation of about 2%. As a result, for the remainder of this study, we do not configure large pages.

Figure 4.1.3 shows the cumulative performance improvement as we go from the default ("out of the box") configuration to doubling the CPU cores for the WPS server. The tunings are shown in a chronological order from left to right in the figure.

It should also be noted that the x3650 server hosting the benchmark's Workload Driver does not have any impact on our performance results because the CPU utilization of this server is less than 5% and the utilization of the only disk used on this server is less than 6%.

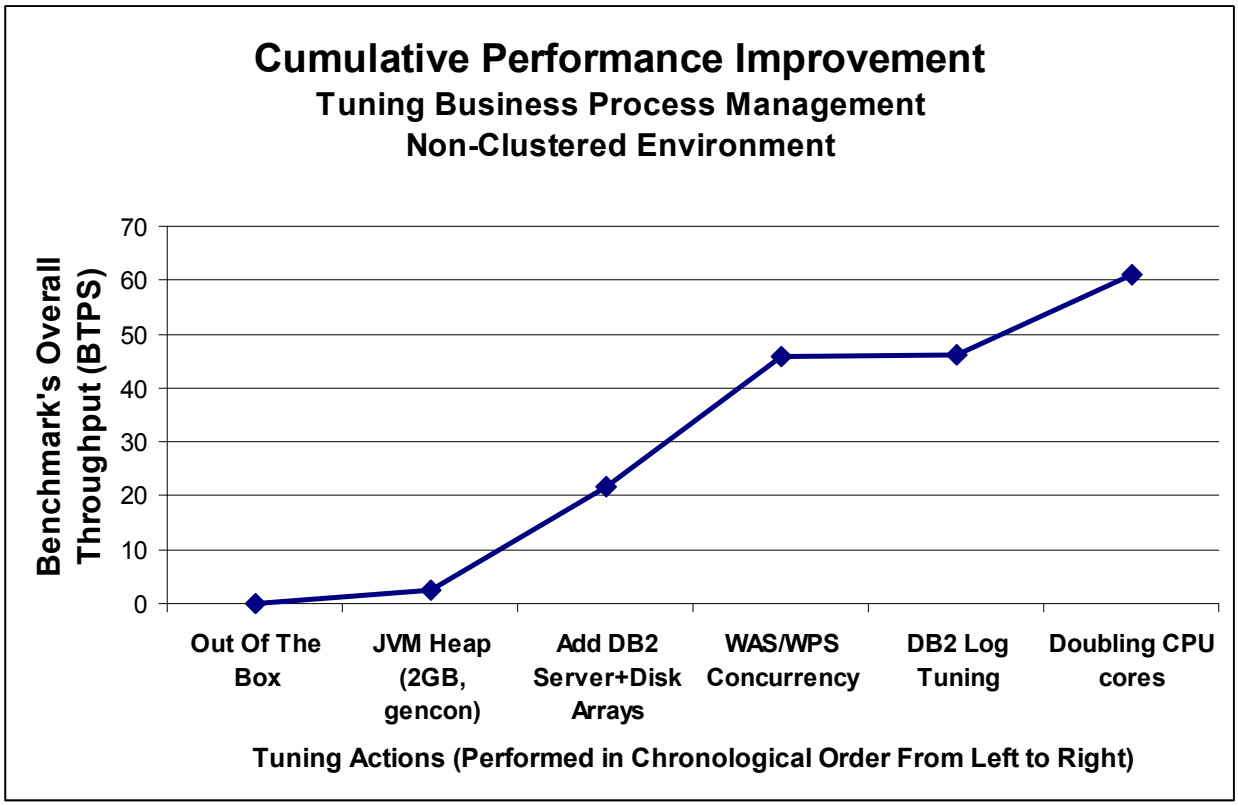

Figure 4.1.3 – Cumulative Impact of BPM Tuning Actions in Non-Clustered Environment

4.2. Tuning Dynamic Operations with WebSphere Virtual Enterprise (WVE)

In this section, we will evaluate a virtual, goals-driven, autonomic environment for SOA applications like the benchmark used in this study. We'll use WebSphere Virtual Enterprise (WVE) to create this environment, which is illustrated in *Figure 4.2.1.*

The dynamic cluster of WPS servers, called "application cluster" in *Figure 4.2.1*, consists of WPS servers on which the benchmark's applications are deployed. These applications include *HandleClaim* and *ClaimServices* applications, denoted as "B" and "S", respectively. In response to an increase in workload demands or an on-going failure to meet a service policy's performance objective, WebSphere Virtual Enterprise's autonomic managers can start an instance of WPS server on the x3850 M2 server or on any of the two *Logical Partitions (LPARs)* on the OpenPower 720 server. Based on what we learn in the previous Section 4.1, each instance of the WPS server has a single disk for transaction log and another single disk for compensation (recovery) log. In *Figure 4.2.1*, the LPARs are virtualized with PowerVM technology, so I/O operations to these (virtual) disks must go through the *Virtual I/O Server (VIOS)*. The LPARs use *shared, capped* virtual CPUs. In general, it is recommended that we configure shared, uncapped virtual CPUs across the LPARs; however, in this study, the number of (virtual) CPUs that each LPAR is allowed to use is capped so we can more accurately measure the workload impact on each LPAR and the performance impact of logical partitioning.

By default, there can only be one WPS server instance maximum per node for the dynamic application cluster.

The service policy in WVE only allows two types of performance goals to be specified – the *average response time* or the *percentile response time* goals. In this study, we have decided to use the *average response time* goal for simplicity. To set a reasonable response time goal, we examine the response

times reported in *Table 4.2.1* for all configurations that we evaluate in this study. We note that the best response time that we are able to achieve is about 14 ms. As a rule of thumb (best practices), the *average response time* goal should be at least 2X higher than the best response time observed on a lightly loaded cluster. Consequently, we set the *average response time* goal in the service policy to be 30 ms. This should allow the autonomic managers to activate additional WPS servers (instances) when the average response time for claim requests exceeds 30 ms (the performance goal). Setting the *average response time* goal too close to 14 ms would never result in additional servers to be started because the autonomic managers would determine that starting additional servers would not improve the capability of meeting the goal.

We specify the service policy to be applicable to HTTP & SOAP requests handled by the benchmark's applications. We also assign the highest priority to this service policy.

We use the default settings for the On-Demand Router (ODR) and its associated autonomic managers.

As described in *Section 2.3*, there are two approaches to cluster the Message Engines (MEs):

- The Message Engines are "local" to the dynamic application cluster
- The Message Engines are in their own cluster (called *MECluster*), separate from the dynamic application cluster

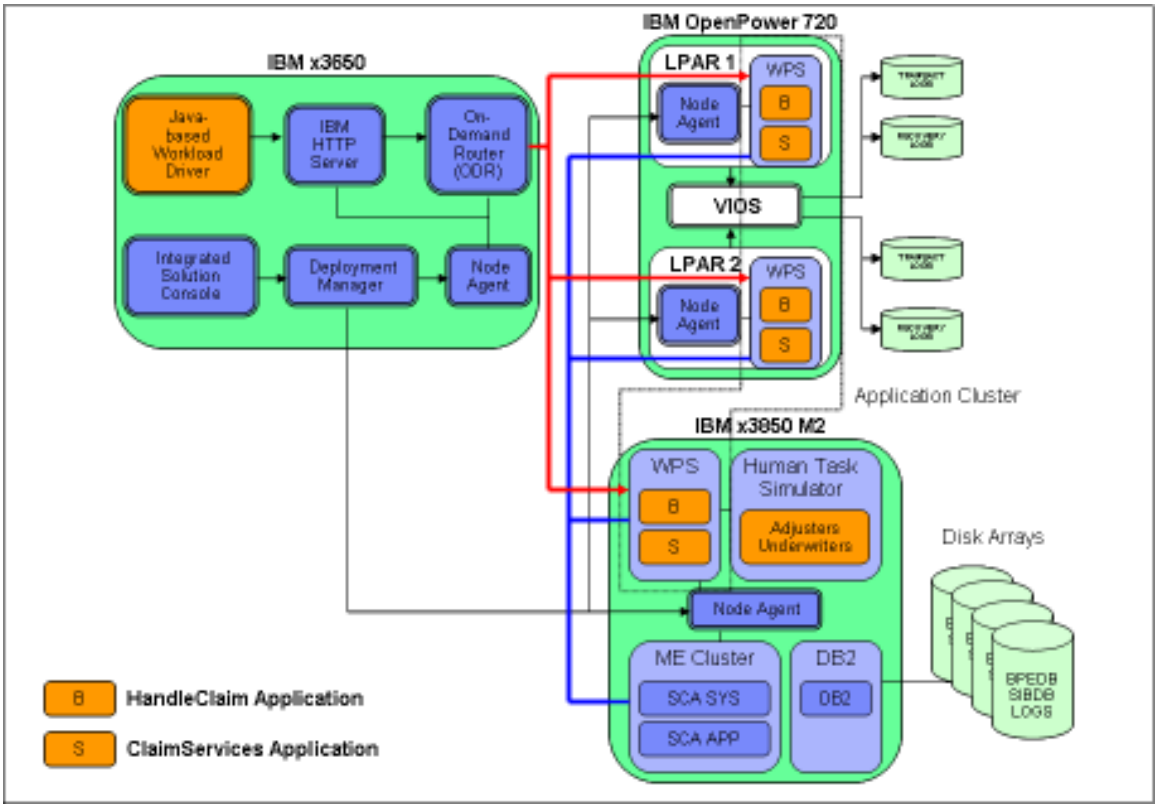

Figure 4.2.1 – Dynamic Application Cluster with Separate Cluster for Message Engines

In this section, we will consider both approaches and see which one performs better. In those configurations where MECluster is configured, it runs on the x3850 M2 server. As such, the MECluster is a static cluster.

In general, for the configurations considered in this section, the physical servers host the following components (see *Figure 4.2.1*):

- IBM x3650 server
  - Benchmark's Java-based Workload Driver
  - WebSphere Deployment Manager
  - IBM HTTP Server (Web Server)
  - WebSphere On-Demand Router (ODR)
  - WebSphere Node Agent

- IBM x3850 M2 server
  - DB2, hosting SIBDB and BPEDB databases (including their logs)
  - Benchmark's *HumanTaskSimulator*, simulating the actions of *Adjusters* and *Underwriters*
  - WPS Server (initial instance – unless otherwise specified), hosting the benchmark's *HandleClaim* and *ClaimServices* applications
  - WebSphere Node Agent
  - ME Cluster (if configured)

- IBM OpenPower 720 (VIOS with 0.5 P5 core, LPAR1 with 1.5 P5 cores, LPAR2 with 2 P5 cores)
  - Virtual I/O Server (VIOS)
  - LPAR1: WPS Server (one instance max) + WebSphere Node Agent
  - LPAR2: WPS Server (one instance max) + WebSphere Node Agent

On the x3850 M2 server, the BPEDB database is hosted on a 24-disk array managed by the IBM System Storage DS3400 controller. Because of the large amount of disk writes, the BPEDB log resides on a separate 12-disk array attached to the IBM ServeRAID MR10M adapter. The SIBDB database and its log are hosted on another 12-disk array attached to the same MR10M adapter. As in *Section 4.1*, each WPS server has two disks, one dedicated for transaction log and one for compensation (recovery) log. For optimal performance, cache mirroring is disabled for the DS3400 controller and write-back caching policy is used for the MR10M controller. In addition, all of the WPS and DB2 tunings discussed in *Section 4.1* are applied to the WVE environment.

For the remainder of this paper, we will denote the maximum number of concurrent insurance claim requests generated by the benchmark's Workload Driver as *MaxTx* and the number of threads used to generate this traffic as simply *Threads*. We will look at two performance measurements: the *Business Transactions Per Second (BTPS)* as reported by the benchmark and the *average response time* for service requests as reported by WebSphere Virtual Enterprise.

Let us first consider the configurations with the Message Engines in a separate cluster (MECluster). *T*he performance results for these configurations are shown in *Table 4.2.1*.

- **Single Node Performance (with separate ME Cluster): *Configuration 5*** in *Table 4.2.1* shows what can be achieved with a single node (the x3850 M2 server) in the dynamic cluster. The dynamic application cluster is put in *Manual Mode*, so that the ODR's autonomic managers do not attempt to activate additional WPS servers on the other nodes – even when the average response time exceeds the performance goal of 30 ms.

  Result: With the Workload Driver using 2 threads to generate a maximum of 60 concurrent requests (MaxTx = 60, Threads = 2), the x3850 M2 is able to handle an average of 28.29 BTPS. The CPU utilization on the x3850 M2 is 83% and the disk subsystem still has quite a bit of bandwidth available (the disk I/O queue depth is less than 0.60). The average response time for the claim requests ranges from 22.85 ms with MaxTx = 20 to 52.74 ms with MaxTx = 60. Setting the max number of concurrent requests higher than 60 introduces some instability: only a few runs are completed successfully as the CPU utilization moves past 90%.

Config	Workload		BTPS	CPU Utilization			Min Response Time (ms)			Max Throughput (BTPS)			Max Disk	
	Max Tx	Threads		Node 1	Node 2	Node 3	Node 1	Node 2	Node 3	Node 1	Node 2	Node 3	Queue Depth	Util
5	20	1	18.23	49.60%	N/A	N/A	22.85	N/A	N/A	20.03	N/A	N/A	0.29	19%
	40	1	25.06	73.50%	N/A	N/A	32.09	N/A	N/A	27.88	N/A	N/A	0.5	28%
	60	1	25.62	75%	N/A	N/A	37.27	N/A	N/A	28.6	N/A	N/A	0.55	30%
	60	2	28.29	83%	N/A	N/A	52.74	N/A	N/A	31.45	N/A	N/A	0.53	28%
5.1	20	1	15.24	35%	45%	55%	18.77	49.41	50.08	10.91	3.59	3.61	0.39	38%
	40	1	20.72	44%	72%	78%	16.02	57.64	55.61	14.08	4.91	5.08	0.55	53%
	60	1	21.88	40%	75%	78%	17.76	66.08	64.83	14.49	5.25	5.35	0.64	61%
	60	2	26.02	58%	81%	85%	19.22	82.99	81.4	17.04	5.98	6.15	0.64	61%
	80	2	30.86	65%	91%	92%	25.51	91.37	89.45	19.62	6.53	7.13	0.76	71%
	80	3	34.29	75%	95%	95%	35.3	103.62	102.78	24.04	6.49	6.87	0.67	68%
	100	3	33.71	77%	92%	92%	40.47	196.22	162.2	22.35	5.01	8	0.78	67%
	160	4	35.56	86%	92%	93%	82.34	263.48	212.12	26.99	4.83	6.67	1.04	70%
5.2	160	4	38.73	86%	72%	N/A	73.05	121.23	N/A	28.68	12.58	N/A	1.24	50%
	160	6	40.47	91%	80%	N/A	68.63	145.3	N/A	29.73	14.1	N/A	1.26	50%
6	20	1	23.55	40%	N/A	N/A	16.18	N/A	N/A	N/A	N/A	N/A	0.38	24%
	40	1	37.36	64%	N/A	N/A	20.28	N/A	N/A	N/A	N/A	N/A	0.66	39%
	60	1	45.73	77%	N/A	N/A	25.88	N/A	N/A	N/A	N/A	N/A	0.95	47%
	60	2	46.55	80%	N/A	N/A	30.38	N/A	N/A	N/A	N/A	N/A	0.96	48%
6.1	20	1	9.4	13%	92%	N/A	14.97	70.41	N/A	N/A	N/A	N/A	0.43	40%
6.2	20	1	11.27	13%	92%	N/A	14.68	54.82	N/A	N/A	N/A	N/A	0.43	42%
6.3	20	1	15.41	15%	55%	N/A	13.6	37.46	N/A	5.22	10.38	N/A	0.51	50%

Table 4.2.1 – Performance Data for Dynamic Cluster Configurations

- **Multi-Node Performance (with separate ME Cluster)**: ***Configuration 5.1*** in *Table 4.2.1* shows that we can achieve higher service rates when we put the dynamic application cluster in *Automatic Mode*. This allows the On-Demand Router's autonomic managers to activate additional servers on the OpenPower 720's LPARs automatically (without human intervention) to help handle the workload generated by the benchmark's Workload Driver. This configuration is illustrated in *Figure 4.2.1*.

  Initially, after we turn on the benchmark's Workload Driver (with MaxTx = 20, Threads =1), the initial WPS server running on the x3850 M2 takes a while to warm up. During this warm-up period, the average response time for service requests exceeds the service goal of 30 ms for more than 30 seconds, as shown in *Figure 4.2.2*. The autonomic managers sense this condition, and generate a run-time task to activate additional WPS servers on the cluster's remaining nodes. The run-time task appears on the Integrated Solutions Console (ISC)'s System Administration View at approximately 2 minutes after the start of the run. Since the application cluster is in Automatic Mode, the run-time task starts immediately without further human intervention after it appears on the ISC. It takes about 2 minutes for the new WPS servers to be activated on the two LPARs on the OpenPower 720 server. Finally, about 4 minutes after the start of the run, the WPS servers on the LPARs start running. Of course, these new servers need time to warm up and that's the reason for the initial high response times recorded on these servers. In the mean time, as the new WPS servers are starting on the LPARs, their Message Engines are starting to run in the ME cluster on the x3850 M2, causing the average response time on the x3850 M2 server to jump higher as well. However, after about 90 seconds, the average response time on the x3850 M2 starts to stabilize and falls to around 18.77 ms – well below the service goal of 30 ms and better than the best average response time in the single-node configuration (*Configuration 5*). This shows the benefit of "off-loading" the workload to new additional servers in the dynamic cluster. However, the LPARs (with 1.5 P5 cores for LPAR1 and 2.0 P5 cores for LPAR2) hosting the new servers are not powerful enough to achieve a response time of less than 30 ms for the service requests sent to them. Because of this, the maximum throughput data collected at each node indicates that the On-Demand Router does route more service requests to the more powerful x3850 M2 server than to the two P5 LPARs, as expected.

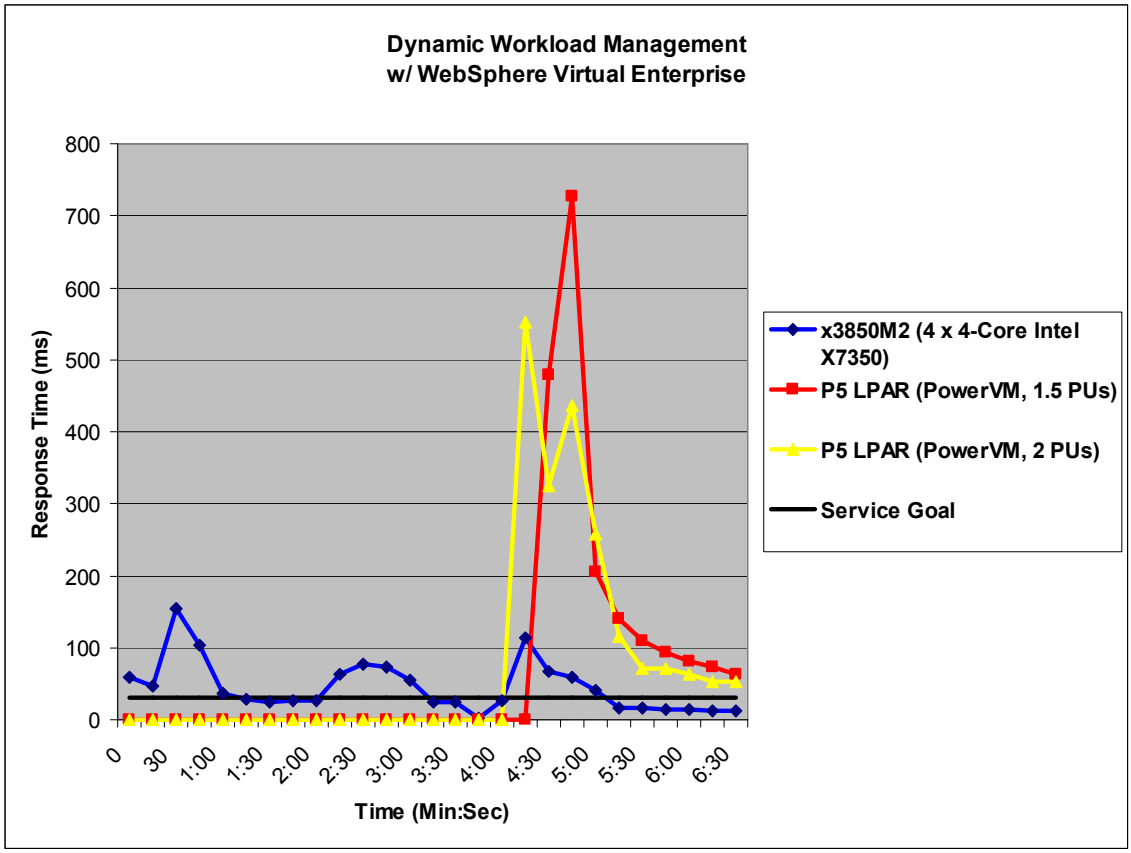

Figure 4.2.2 – WebSphere Virtual Enterprise's Autonomic Managers activating additional WPS servers

Once everything settles down, the best benchmark throughput we can get for this multi-node configuration is 35.56 BTPS – a 25% improvement over the single-node configuration. In addition, as can be seen in *Table 4.2.1*, the multi-node configuration provides more stability at much higher workloads (up to MaxTx = 160, Threads = 4) than the single-node configuration. At the highest workloads (MaxTx = 160, Threads = 4), the CPU utilization is 92%, 93%, and 86% for LPAR1, LPAR2, and x3850 M2 servers, respectively, indicating that we are running out of CPU bandwidth on the LPARs. The disk subsystem is still OK, with the max I/O queue depth of 1.04 and the disk with the highest utilization (70%) being the compensation log disk in LPAR2.

Figure 4.2.6 shows the throughput contributed by each node in the dynamic cluster. As the workload increases, the throughput at the LPARs begins to level off, running out of CPU bandwidth. However, with 16 processor cores, the x3850 M2 just keeps going as the ODR sends more requests to it.

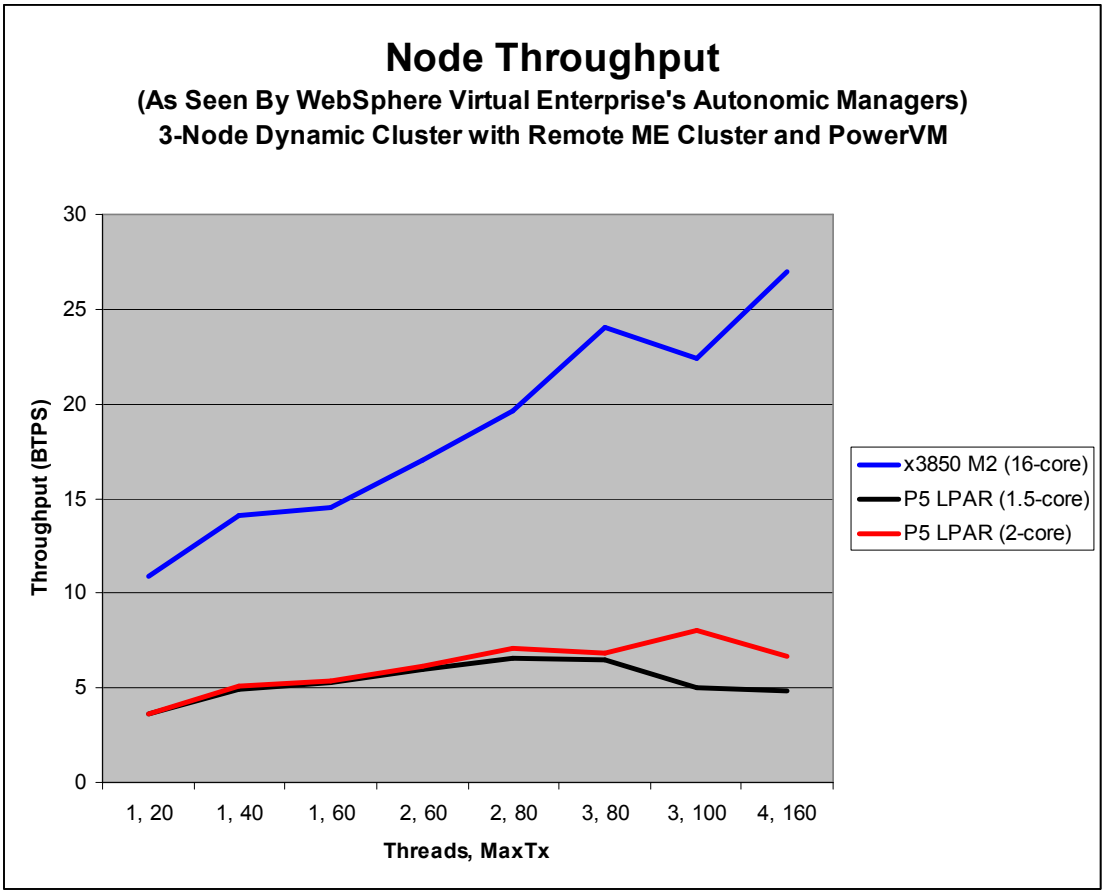

Figure 4.2.6 – Node throughput as reported by WebSphere Virtual Enterprise

- **Multi-Node Performance (with separate ME Cluster and without PowerVM)**: ***Configuration 5.2*** in *Table 4.2.1* shows the benchmark throughput when we effectively remove the PowerVM virtualization layer from the OpenPower 720 server. In the previous *Configuration 5.1*, since we are running out of CPU bandwidth and the disk utilization is pretty high on the LPARs with MaxTx = 160 and Threads = 4, removing the virtualization layer might result in higher benchmark throughput. To do this, we only configure a single LPAR on the OpenPower 720 server, assign all CPUs (4 cores) and memory to this single LPAR, and remove the Virtual I/O Server (VIOS). All I/O operations now go directly to the disk subsystem without going through the VIOS, reducing the I/O latency. This configuration is illustrated in *Figure 4.2.3*.

  Result: As can be seen in *Table 4.2.1*, this configuration is now able to support up to 6 threads with MaxTx = 160, and the overall benchmark throughput is now 40.47 BTPS – a 14% improvement over the previous *Configuration 5.1*. The CPU utilization on the x3850 M2 server now moves past 90%, but the disk subsystem still has available bandwidth as the disk I/O queue depth is 1.26 and the highest disk utilization is only 50%. If we look at the data for MaxTx = 160, Threads = 4 and compare it against the data for the same workload in the previous configuration, we see that the average response time on the single 4-core LPAR is less than half the response times observed on the 1.5-core LPAR1 and 2-core LPAR2 in the previous configuration. While the maximum throughput observed at the single 4-core LPAR is about 10% higher than the sum of maximum throughput for both LPAR1 and LPAR2 in the previous configuration, it is interesting to note that the performance of the WPS server running on the x3850 M2 is better too (i.e. lower average response time, higher maximum throughput). This indicates that a single 4-core LPAR can “off-load” more workload from the x3850 M2 than both the 1.5-core LPAR1 and 2-core LPAR2 can in the previous configuration. Later on in this paper, we will measure the performance impact of the Virtual I/O Server (VIOS) more accurately.

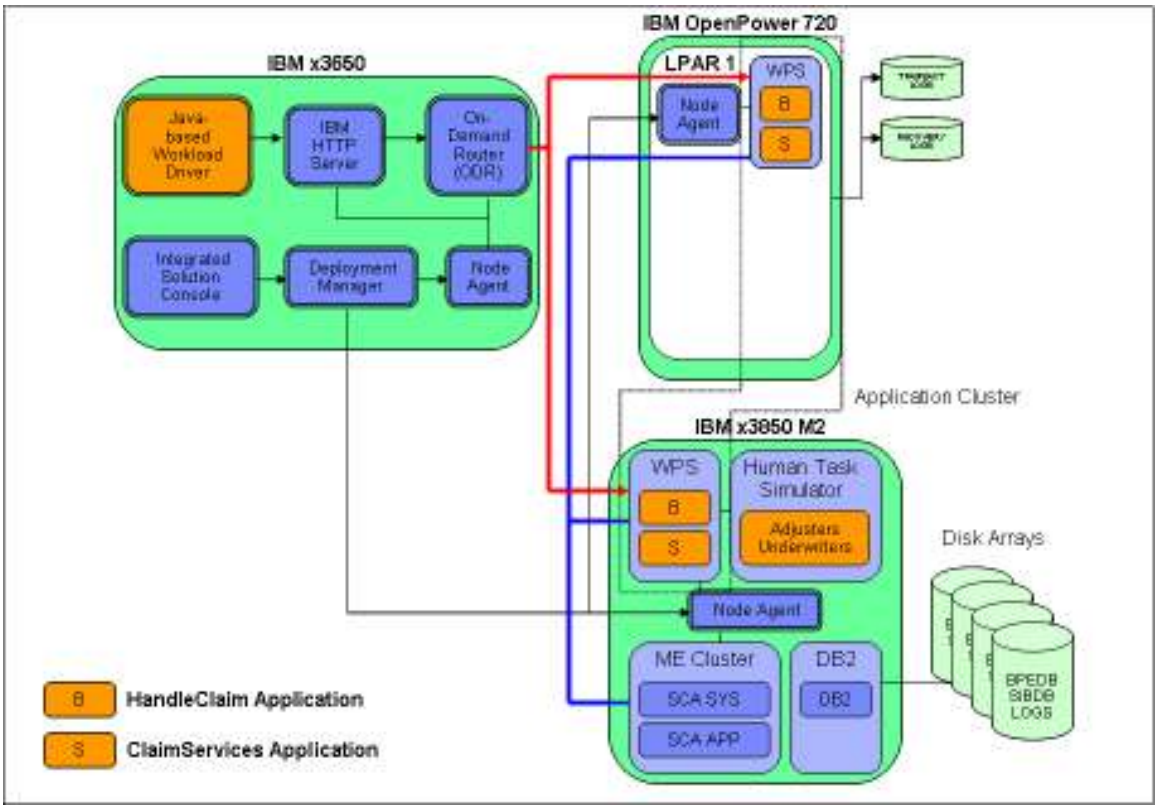

Figure 4.2.3 – Dynamic Application Cluster with Separate ME Cluster but without PowerVM

- **Single-Node Performance (with Local Message Engines)**: ***Configuration 6*** in *Table 4.2.1* shows the performance data when using only a single node (x3850 M2) with "local" Message Engines running in the same dynamic cluster as the WPS servers.

  Result: The data indicates that, for this single-node configuration with "local" Message Engines, the benchmark throughput is much higher, and the average response time is much lower, than in *Configuration 5* where we have "remote" Message Engines located in a separate cluster. Depending on the workload level, this extra latency in going to Message Engines in a remote cluster results in 30% to 60% drop in the benchmark throughput and 25% to 40% jump in the average service response time. Consequently, having the Message Engines running locally in the same cluster with the applications is faster in the single-node configuration, but it severely limits the scalability of a multi-node configuration – a fact that will be very obvious when we consider the next configuration.

- **Multi-Node Performance (with Local Message Engines)**: ***Configuration 6.1*** in *Table 4.2.1* shows the performance data obtained for a 2-node application cluster with "local" Message Engines. This application cluster is illustrated in *Figure 4.2.4*. When we start the Workload Driver, the initial WPS server runs on a 2-core P5 LPAR. As the CPU utilization on this LPAR quickly rises past 90%, and the average service response time is way above the service goal of 30 ms, the autonomic managers generate a run-time task to activate the remaining node (the x3850 M2) in the cluster. However, even after a new WPS server is started on the x3850 M2, most (approximately 80%) of the workload still goes to the initial node (the 2-core P5 LPAR), leaving the second node (the x3850 M2) pretty much under-utilized – the CPU utilization on the x3850 M2 is only 13%. The benchmark's overall throughput (9.4 BTPS) is much smaller than in the case with "remote" Message Engines (35.56 BTPS). This is because of a design limitation for the Service Integration (SI) buses: if a Message-Driven Bean (MDB) has a local ME available, it

is always forcibly "bound" to it, even if the ME is inactive. Since there can only be one ME instance active at any given time, only one WPS server can have active instances of that MDB. In other words, all but one of the WPS servers in the cluster is essentially in stand-by. This imposes serious limitation on the overall scalability of the dynamic cluster if we have long-running business processes or asynchronous SCA invocations. As a result, for multi-node dynamic clusters, we recommend configuring the Message Engines in a separate cluster. In [4], the WebSphere clustering topologies are classified into three types:

- "Bronze" topology where all components (WPS applications, Message Engines, CEI) are run in a single cluster
- "Silver" topology where there are two separate clusters: one for WPS applications and CEI and one for the Message Engines
- "Gold" topology where there are separate clusters for WPS applications, CEI, and Message Engines

Since the benchmark used in this study does not generate many events, the "Silver" topology is the most suitable topology. *Figure 4.2.5* shows the impact of messaging infrastructure clustering on the benchmark throughput.

In this *Configuration 6.1*, the WPS server running on the 2-core P5 LPAR is the initial node, so with local Message Engines, most of the traffic goes there, and the benchmark throughput is only a meager 9.40 BTPS. What if we made the x3850 M2 the initial node? In this case, most of the traffic would be routed to the x3850 M2, and since it is much more powerful than the 2-core P5 LPAR, the benchmark's throughput is 34.62 BTPS – almost *4 times higher* than in the case of the 2-core P5 LPAR being the initial node (with the exact same settings for the Workload Driver). The CPU utilization is 65% and 9% on the x3850 M2 and the P5 LPAR, respectively, confirming that most of the traffic is indeed routed to the initial x3850 M2 node.

- **Virtual I/O Server (VIOS) Impact**: We know that the VIOS introduces additional latency to disk I/O operations. In ***Configuration 6.2***, we remove the VIOS partition, so disk writes to the transaction and compensation logs no longer go through VIOS. At the same workload driver settings (MaxTx = 20, Threads = 1), the data in *Table 4.2.1* indicates that removing the VIOS reduces the average service response time from 70.41 ms to 54.82 ms – a 22% improvement over the configuration with VIOS. Let us take a closer look at the transaction and compensation log disks which are attached to the P5 LPARs. *Table 4.2.2* shows the disk performance statistics for *Configuration 6.1* (with VIOS) and *Configuration 6.2* (without VIOS). The number of disk writes and the amount of written log data are similar in both configurations, as expected. However, *without* VIOS, we are able to obtain:
  - Higher overall throughput for the benchmark (11.27 BTPS vs. 9.40 BTPS)
  - Lower disk I/O queue depth
  - Lower I/O service times
  - Low disk utilization (as seen from Linux)

  However, it should be noted that PowerVM also allows us to configure the log disks as *dedicated* I/O devices to the LPARs so disk writes to those disks do not have to go through VIOS. We will look at the performance of the disk subsystem in more detail later in the paper.

- **Adding More CPUs (Or Using Shared, Uncapped LPAR Mode)**: With the initial WPS Server running on a 2-core P5 LPAR, most of the workload traffic goes there, pushing the CPU utilization on the LPAR very close to 100%. ***Configuration 6.3*** in *Table 4.2.1* shows what we are able to get when we assign more P5 cores to the LPAR. This is effectively the same as if we configure the LPAR to run in *shared, uncapped* mode where the LPAR could use the remaining (idle) processor cores on the OpenPower 720 server if needed. Indeed this is one of the key advantages of PowerVM as we can initially configure an LPAR with a small number of processor cores, and then as workload demand increases, this LPAR can grow to the full size of the physical server. In this configuration, the LPAR now has a total of 4 P5 cores, and as a result,

the CPU utilization drops from 92% to 55% while the average service response time on this LPAR drops from 54.82 ms to 37.46 ms – given the same settings for the Workload Driver.

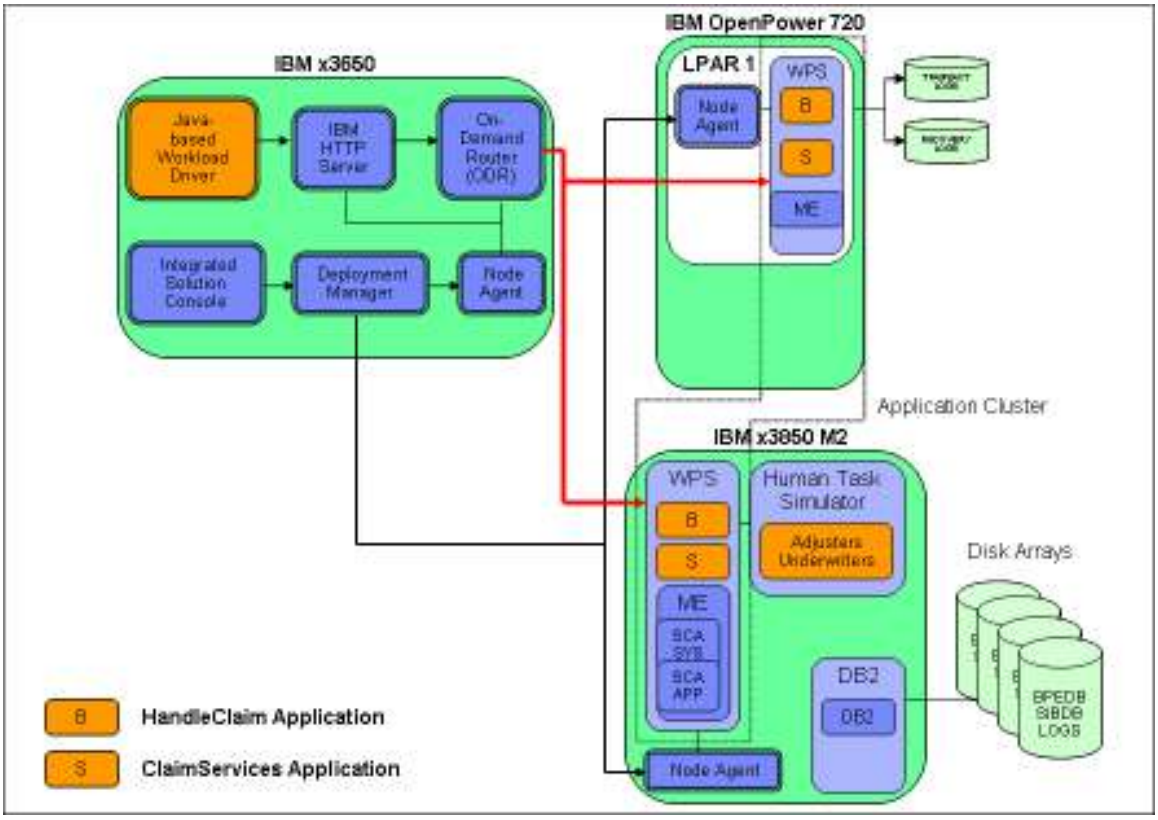

Figure 4.2.4 – Dynamic Application Cluster with Local Message Engines

Config	Max Tx	Threads	BTPS	Server	Data Type	Disk Type (Single Disk)	writes/sec	wKB/sec	avgrq-sz (sectors)	avgqu-sz	await (ms)	svctm (ms)	util
6.1	20	1	9.4	2-core P5 LPAR	TX LOG	VIOS	118	554	9.41	0.41	3.44	3.42	40%
					REC LOG	VIOS	84	424	10.14	0.43	5.13	4.08	40%
6.2	20	1	11.27	2-core P5 LPAR	TX LOG	No VIOS	112	495	8.89	0.37	3.43	3.32	37%
					REC LOG	No VIOS	84	416	9.1	0.26	2.93	2.86	25%

Table 4.2.2 – Impact of VIOS

- **Disk I/O Performance**: *Table 4.2.3* provides the disk I/O statistics for *Configuration 5.1* (3 nodes: 1.5-core P5 virtualized LPAR, 2-core P5 virtualized LPAR, 16-core x3850 M2) and *Configuration 5.2* (2 nodes: 4-core P5 LPAR without VIOS, 16-core x3850 M2). The data is provided for both light workload (MaxTx = 20, Threads = 1) and heavy workload (MaxTx = 160, Threads = 4). Both configurations have Message Engines in a separate cluster. The data indicates that, for an SOA application such as the benchmark used in this study, from a performance perspective, it is better to use the entire OpenPower 720 server as a single partition without PowerVM than configuring LPARs with VIOS. In addition, we note the following from the data:

    - For the disks attached to the virtualized LPARs, the extra latency in going through the Virtual I/O Server (VIOS) results in higher I/O service times and higher disk utilization (as seen from Linux). With VIOS, the I/O service times can reach as high as 15 ms, whereas without VIOS, the I/O service times are always less than 4 ms. Similarly, the disk

utilization data (collected at the operating system level) can reach as high as 70% with VIOS while it always stays below 50% without VIOS. However, the disk I/O queue depth in all cases is less than 1, indicating that this is strictly an I/O latency issue and that the disk subsystems are not really the performance bottleneck in the configurations that we consider in this study.

- Even under the heaviest workload conditions considered in this study, there is still quite a bit of available bandwidth left on the disk arrays: the disk I/O queue depth is mostly less than 1 and the utilization is mostly less than 50%. Even under the heaviest workload, the queue dept is only 1.24.
- Data writes to the BPEDB database tend to be "bursty" in nature (based on the disk I/O queue depth and the disk utilization data); in contrast, data writes to the SIBDB database and to the logs (BPE, transaction, compensation) occur evenly over time (i.e. the disk I/O queue depth data match well with the disk utilization data).
- Data writes to the disk arrays are in the sub-millisecond range, indicating that write-back caching is in effect and very effective, especially since there is not much I/O waiting in the queues. However, the disk arrays attached to the MR10M controller deliver better performance (in terms of I/O service times) because its cache is located much closer to the server than in the case of the DS3400 controller where the controller is located remotely from the server over the fiber links.

- **Tuning the On-Demand Router (ODR)**: The WVE test team recommends that the JVM heap size for the ODR should be at least 1024 MB and that the garbage collection policy should be *Generational Concurrent (gencon)* with the *noAdaptiveTenure* setting. However, since the service request rates in this study are fairly small (less than 100), these ODR tunings do not make much difference in the benchmark's overall throughput. For this study, the default JVM heap configuration for the ODR works just as well.

- **Workload Driver**: The benchmark's Workload Driver, the IBM HTTP Server, the On-Demand Router, and the WebSphere Node Agent are hosted on the x3650 server. The CPU utilization of this server is less than 5% and the utilization of the only disk used on this server is less than 6%, indicating that this server does not have any negative impact on our performance results.

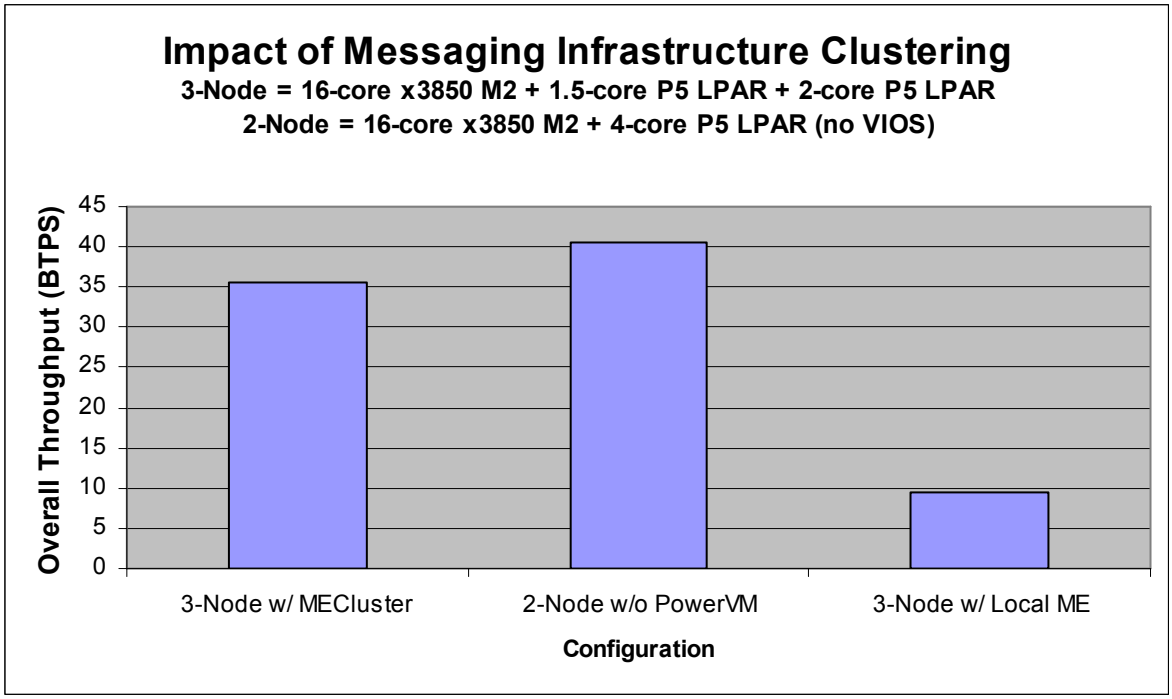

Figure 4.2.5 – Impact of Message Engine Clustering on Overall Throughput

Config	Max Tx	Threads	BTPS	Server	Data Type	Disk Type	writes/sec	wKB/sec	avgrq-sz (sectors)	avgqu-sz	await (ms)	svctm (ms)	util
5.2	160	4	38.73	16-core x3850M2	BPE LOG	12-disk MR10M	1177	9261	15.74	0.06	0.06	0.05	6%
					SIBDB	12-disk MR10M	1451	6634	9.14	0.06	0.05	0.04	6%
					BPEDB	24-disk DS3400	1555	6656	8.56	1.24	0.8	0.32	50%
					TX LOG	12-disk DS3400	254	1121	8.52	0.17	0.68	0.68	17%
					REC LOG	12-disk DS3400	186	891	9.58	0.12	0.64	0.64	12%
				8-core P5	TX LOG	SCSI disk	143	646	9.05	0.47	3.31	3.28	47%
					REC LOG	SCSI disk	135	651	9.68	0.46	3.42	3.16	43%
	20	1	26.36	16-core x3850M2	BPE LOG	12-disk MR10M	1509	8257	10.94	0.07	0.04	0.04	7%
					SIBDB	12-disk MR10M	1978	8061	8.15	0.08	0.04	0.04	8%
					BPEDB	24-disk DS3400	1014	4290	8.96	0.71	0.7	0.33	34%
					TX LOG	12-disk DS3400	184	808	8.78	0.11	0.61	0.61	11%
					REC LOG	12-disk DS3400	128	607	9.5	0.08	0.61	0.61	8%
				8-core P5	TX LOG	SCSI disk	129	577	8.95	0.42	3.3	3.24	42%
					REC LOG	SCSI disk	101	485	9.6	0.31	3.1	2.96	30%
5.1	160	4	35.56	16-core x3850M2	BPE LOG	12-disk MR10M	972	8491	17.47	0.07	0.07	0.07	7%
					SIBDB	12-disk MR10M	1283	6076	9.47	0.08	0.06	0.06	8%
					BPEDB	24-disk DS3400	1344	5980	8.9	1.04	0.77	0.34	46%
					TX LOG	12-disk DS3400	226	1003	8.55	0.13	0.58	0.57	13%
					REC LOG	12-disk DS3400	183	879	9.6	0.12	0.66	0.66	12%
				3.5-core P5	TX LOG	VIOS disk	67	330	9.62	0.53	10.85	10.77	57%
					REC LOG	VIOS disk	42	228	10.62	0.64	15.83	14.56	65%
				4-core P5	TX LOG	VIOS disk	78	357	9.19	0.46	5.97	5.79	45%
					REC LOG	VIOS disk	52	288	11.02	0.74	14.24	13.47	70%
	20	1	15.24	16-core x3850M2	BPE LOG	12-disk MR10M	1067	5558	10.41	0.04	0.04	0.04	4%
					SIBDB	12-disk MR10M	1481	5998	8.1	0.07	0.05	0.05	7%
					BPEDB	24-disk DS3400	494	2064	8.35	0.3	0.61	0.37	18%
					TX LOG	12-disk DS3400	91	395	8.72	0.04	0.46	0.46	4%
					REC LOG	12-disk DS3400	65	308	9.42	0.04	0.56	0.56	4%
				3.5-core P5	TX LOG	VIOS disk	52	275	10.32	0.25	4.77	4.66	24%
					REC LOG	VIOS disk	39	196	9.83	0.37	9.34	9.04	36%
				4-core P5	TX LOG	VIOS disk	70	324	9.31	0.28	3.95	3.87	27%
					REC LOG	VIOS disk	43	213	9.93	0.39	9.07	8.89	38%

Table 4.2.3 – Performance statistics for disk subsystems

5. Conclusions

Based on the data analysis in Section 4, we can state the following conclusions:

- **Business Process Management (BPM) tuning in a non-clustered environment**:
    - As with other WebSphere facets, the WAS JVM heap configuration is important. For the BPM scenarios considered in this study, the default JVM heap size is too small; at least 2048 MB is recommended. In some cases, increasing the JVM heap size from 2048 MB to 4096 MB can result in a 20% performance gain. The WAS JVM heap size should also be adjusted accordingly after adding resources, such as processors, to the server or partition hosting the WPS server to accommodate more workload. It is also recommended to use the Generational Concurrent (gencon) garbage collection policy as it is most suitable for Web service transactions.
    - With a large volume of disk write traffic to the SIBDB database, BPEDB database, and BPE log, it is recommended that they are hosted on physical disk arrays, such as those connected to the IBM ServeRAID MR10M and IBM System Storage DS3400 controllers used in this study. If the disk arrays are attached to a host controller through a fiber-channel interface (such as the DS3400 controller used in this study), it would be better to configure a separate disk array for the BPEDB database because, in our testing, the average disk I/O queue depth becomes greater than 1.0 under peak workload conditions even when we dedicate a 24-disk array to BPEDB.
    - For best performance, cache mirroring and cache write-through in the disk array controllers should be disabled. However, this is a classic trade-off between performance and reliability / availability.
    - For data writes to WPS databases and logs, the IBM ServeRAID MR10M disk subsystem delivers better performance than the IBM System Storage DS3400 disk subsystem. This can be attributed to the fact that the write-back cache on the MR10M is much closer to the WPS server than the cache in the DS3400 which is located remotely from the server over the fiber links.
    - It is better to have the DB2 component run on a separate physical server from the WPS server, especially when there are multiple WPS servers in a cluster configuration.
    - Business process modeling applications, such as the one used in our benchmark, are generally processor-intensive. Adding more processors to the WPS Server would certainly increase the benchmark throughput – provided that some J2C Connection Factories and Activation Specifications would also be increased to accommodate the higher number of processors as recommended by the WebSphere Performance Team [13].
    - The transaction and compensation logs for each WPS server should be hosted on separate disks if physical disk arrays are not available. These disks (or disk arrays) should be local to the WPS server. Hosting the transaction and compensation logs on a remote server, even on disk arrays, would result in very low performance due to the large amount of disk writes involved and the latency requirements. Logical disk striping through the Linux Volume Manager is also not very effective in handling the log traffic.
    - The WAS and WPS concurrency parameters, such as thread pools (e.g. default, web container), connection pools (for BPEDB, CEI ME, SIBDB System, and SIBDB BPC data sources), J2C connection factories (for BPECF, BPECFC, HTMCF), and J2C Activation Specifications, should have appropriate values based on the number of processors (as

seen by the operating system). In this study, giving these parameters appropriate values essentially doubles the benchmark's overall throughput (see *Section 4.1*).

- DB2 log tunings, such as the maximum storage for lock list, log buffer size, log file size, etc., yield only minor (less than 2%) improvement in the overall business transaction rate.

- The Linux Large Page Support is not effective for BPM scenarios.

- **Tuning Dynamic Operations in a WebSphere Virtual Enterprise (WVE) Environment**:

  - WebSphere Virtual Enterprise with its On-Demand Router (ODR) and associated autonomic managers create a virtual, goal-driven, autonomic cluster environment for SOA applications that can optimize resource usage and provide the capability of responding in real-time to spikes in workload demands (without human intervention if *Automatic Mode* is used). This capability includes the activation of new application server instances on available physical servers to handle the peak workload.

  - Through the use of Java, the application servers are really separated from the physical server hardware. In our setup, application server instances are activated on different server hardware platforms (e.g. Power and x86 servers) to handle the same stream of service requests. A single DB2 server instance on an x86 server can support multiple WPS server instances on that x86 server as well as on other Power servers.

  - For messaging infrastructure clustering, having the Message Engines running locally in the same dynamic cluster as the WPS applications is faster, but it severely limits the overall scalability of a multi-node configuration. For multi-node configurations, it is recommended that the Message Engines run in their own cluster, separate from the application cluster. Although desirable, it is not necessary for the Message Engine cluster to run on different physical server(s) from those hosting the application cluster(s) to achieve good scalability. In our study, the Message Engine cluster runs on the x3850 M2 server, which hosts DB2 and also part of the dynamic application cluster.

  - Care should be taken when setting the performance goal in the service policy. WVE V6.1 only supports response-time-related service goals. Setting the service goal too close to the minimum service response time on a lightly loaded configuration would never result in additional servers to be activated because the autonomic managers would determine that starting additional servers would not improve the capability of meeting the service goal anyway. In our study, we set the service goal to be 2X the minimum service response time.

  - It takes at least 3 to 4 minutes from the time a service goal is considered *breached* until additional servers are started and running. Even then, it may take a few more minutes for the new servers to "warm up" and be able to handle service requests at optimal rates.

  - From a pure performance perspective, if a Power server is used to host WPS server(s), it is better to configure a single LPAR with all the resources on the physical server and use dedicated disks for transaction and recovery logs (as opposed to configuring multiple LPARs with PowerVM). For the benchmark in this study, a single LPAR with all four P5 cores on an OpenPower 720 server delivers 14% more business transactions per second than two virtualized LPARs on the same server. Alternatively, with PowerVM, we can start small and put the LPAR in *shared, uncapped* mode, so the LPAR can grow into the full size of the physical server as workload demand increases.

  - The Virtual I/O Server (VIOS) results in higher I/O service times and higher disk utilization. We have seen the virtual I/O service time as high as 3X the dedicated I/O service time. The VIOS overhead is also responsible for 20% increase in the average service response time in this study. However, PowerVM does allow us to configure the

disks as dedicated I/O devices to the LPARs so I/O operations to these disks do not go through VIOS. For WPS servers, if we need to use PowerVM for availability and other virtualization benefits, we should host the transaction and compensation logs on dedicated disks to each LPAR.

- As we can see from the performance data in this study, PowerVM does have some performance overhead. However, its features and benefits are quite many, among them:

  - Server consolidation of existing workloads onto Power servers, thereby addressing the challenges of reducing space, power, and overall life-cycle costs
  - High-Availability solutions with Live Partition Mobility (available for POWER6 processors or later only)
  - WebSphere Virtual Enterprise can work with server provisioning solutions, such as the Tivoli Provisioning Manager (TPM) or Tivoli Intelligent Orchestrator (TIO), to dynamically allocate new (physical or virtual) servers on which application server instances can be activated to handle unexpected surges in workload demands
  - Low-cost application security and isolation with EAL4+ certification
  - Far better granularity than hardware partitioning techniques (e.g. multiple LPARs can be allocated within a single processor to a granularity of $1/100^{th}$ of a processor, with $1/10^{th}$ of a processor being the minimum for an LPAR)
  - I/O can be virtualized and reconfigurable, and a mix of shared and dedicated I/O is supported across LPARs
  - In shared, uncapped mode, an LPAR can grow to the full size of the physical server as long as resources are available to handle peak workload demands

  The last two features – the support of dedicated I/O and shared, uncapped LPAR mode – can help reduce some performance overhead of PowerVM in our setup. For example, as discussed previously, for WPS servers, the transaction and compensation logs should be hosted on dedicated disks to each LPAR. Instead of adding more processor cores to an LPAR, we can start small and put the LPAR in *shared, uncapped* mode where it can use additional available processors on the physical server as necessary.

- For the business process scenarios considered in this study, the benchmark's Workload Driver, the IBM HTTP Server, and the On-Demand Router do not consume much CPU or I/O resources at run time. The default configurations for the IBM HTTP Server and the On-Demand Router are found to be more than adequate to support the request rates of less than 100 requests per second in this study.

References

[1] Services Oriented Architecture (SOA) Entry Points, http://www-306.ibm.com/software/solutions/soa/entrypoints/index.html?S_TACT=107AG01W&S_CMP=campaign

[2] WebSphere Application Server, http://www-306.ibm.com/software/webservers/appserv/was/

[3] WebSphere Process Server: IBM's New Foundation for SOA, http://www.ibm.com/developerworks/websphere/library/techarticles/0509_kulhanek/0509_kulhanek.html

[4] Clustering WebSphere Process Server V6.0.2, Part 1: Understanding the topology, http://www.ibm.com/developerworks/websphere/library/techarticles/0704_chilanti/0704_chilanti.html

[5] IBM Power 570, http://www-03.ibm.com/systems/power/hardware/570/

[6] IBM POWER6 Microarchitecture, IBM Journal of Research and Development, http://www.research.ibm.com/journal/rd/516/le.html

[7] IBM System x3850 M2, http://www-07.ibm.com/systems/includes/content/x/pdf/XSD03019USEN.pdf

[8] IBM System x3850 M2 Enterprise Server's X4 Technology, http://www-03.ibm.com/systems/x/hardware/enterprise/x3850m2/x4/info.html

[9] The Basic Web Services Stack: IT Web Services: A Roadmap for the Enterprise, by Alex Nghiem, Prentice Hall (October 8, 2002)

[10] SOABench 2005, Version 0.18, Document Owner: Andrew Schofield, Internal IBM Confidential Document (April 18, 2006)

[11] IBM OpenPower 720, http://www.ibm.com/systems/p/hardware/openpower/720/index.html

[12] SOA Performance on Linux: Services and Reuse Entry Point, by Steve Dobbelstein, Khoa Huynh, Vivek Kashyap, and Mark Peloquin, Internal IBM Document (June 2008)

[13] WebSphere Process Server (WPS) 6.1.0, WebSphere Enterprise Service Bus (WESB) 6.1.0, WebSphere Business Monitor (Monitor) 6.1.0, WebSphere Adapters (WA) 6.1.0 Performance Report, by WebSphere Process Server, Enterprise Service Bus, Adapter, and Monitor Performance Teams, Internal IBM Document (April 2008)

IBM Systems and Technology Group
3039 Cornwallis Road
Research Triangle Park, NC 27709

Produced in the USA
06-08