z/OS® Workload Manager

Managing CICS® and IMS™ Workloads

INSIDE WLM Issue No. 2

Robert Vaupel
Senior Technical Staff Member
IBM Böblingen Development Laboratory
Schönaicherstr. 220
Tel.: +49(0)-7031-16-4239
Email: vaupel@de.ibm.com

IBM Copyright © 2014

March 19, 2014

Contents

IBM. Copyright © 2014

List of Figures

IBM Copyright © 2014

1. Introduction

This document gives an introduction how WLM manages CICS and IMS workloads. We will explain how WLM recognizes CICS and IMS transactions and how these results are used to manage CICS and IMS workloads. An important part is how installations can make best use of this methodology and how they can setup goals for CICS and IMS workloads. At the end we will take a look at advanced options for managing CICS and IMS and how installations can influence WLM management.

In this document all examples mention CICS. For all practical purposes the results are the same for IMS and the discussed algorithms and methodologies are the same for both subsystems.

1.1. Units of Work

One of the big strengths of z/OS Workload manager is its ability to manage units of work which enter the z/OS system perform an end user task and leave the system. WLM is able to distinguish different kinds of units of work:

- The most basic construct is the address space. An address space is recognized as a unit of work when it starts and the unit of work is completed when the address space ends. There is no more information available of the work being executed in the address space. Therefore WLM manages address spaces by collecting data of its resource consumption and delays which they encounter while they execute.
- More information exists for Batch work. Batch work re-uses already pre-started address spaces. The unit of work is the Batch job and it is possible to determine its execution time when the batch job starts to run in the address space until it ends. This allows managing the batch work not only by its observed using and delay information but also towards an expected response time.

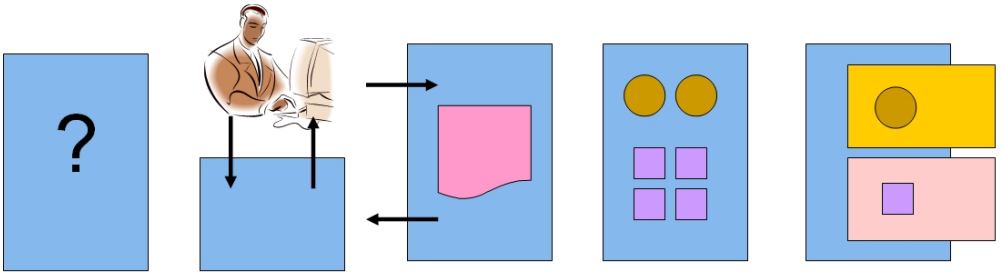
  Similar to batch are requests being processed by Advanced Program to Program Communication (APPC). APPC also has initiators address spaces started in advance which receive incoming work and tell WLM when the new work requests start and end.
- The next step of more detailed information about work being executed is available for TSO and Unix System Service (OMVS) users on z/OS. An address space is created for each TSO and OMVS user. A unit of work executed for a TSO user is now the work request started when the TSO user presses the ENTER button until the result of the request is returned to the terminal. The user address space uses a system request (SYSEVENT) to tell WLM when the user presses the ENTER button and when the work request is completed. This allows WLM to encapsulate very short running requests as well as long running requests and enables WLM to distinguish and manage short from long running work of the same set of end users.
- The most interesting type of end user requests are those received from outside the system by transaction manager and database systems. Transaction manager and databases provide a work receiver and consumer infrastructure and guarantee that the end user requests adhere to transaction processing attributes. The majority of work running on large mainframe systems is processed by such kind of middleware or subsystems. WLM provides two sets of application programming interfaces which allow the middleware to tell WLM when work requests start, end and in which subsystem address space they execute:

  1. The first set of interfaces are named subsystem work manager service and they described in detail in [13]. These interfaces are exploited by Customer information Control System (CICS)

and Information Management System (IMS). These interfaces tell WLM when units of work start and end. They allow assigning the requests to WLM service classes and allow WLM to associate the address spaces to the transaction types.

2. The second set of interfaces allows encapsulating individual work requests on z/OS. They introduce a new construct an enclave on the z/OS system. An enclave represents a unit of work and is associated with a WLM service class. The work manager address spaces then use WLM programming interfaces to tell WLM which executable unit is associated with the end user request. This allows WLM to not only determine the execution time of the enclaves but also to manage them independently[1] from address spaces on z/OS.

In this document we will fully concentrate on subsystem work manager services which are exploited by CICS and IMS. The document will also use examples for CICS but most of them are applicable in the same fashion for IMS.



	Processes	Online User	Batch Work	Online Transaction Processing	
	Address Space Process	TSO/OMVS user	Batch/APPC job	CICS/IMS	Websphere DB2, SAP, etc
What does WLM know about it?	An address space which is using resources (or waiting on resources)	An address space which tells the system when a transaction starts and ends	Start and end of job describe the live of a transaction	Address spaces tell the system which transactions are being processed	Enclaves are a mean to encapsulate a transaction.
Unit of Work (Transaction)	Unknown	Press ENTER Terminal Out	Begin and End of Job	Informed by CICS and IMS	Enclave created by subsystem
Duration (Life Time)	"Indefinite"	Typically short Some longer	Typically long Some short	Typically very short	Short, medium, long
Base for WLM Management	Execution States	Response Time	Response Time or Execution States	Response Times	Response Time or Execution States

Figure 1.1.: Units of Work on z/OS

Figure 1.1 summarizes the different types of units of work on z/OS including the knowledge WLM is able to obtain about them. The column "base for WLM management" refers to the data which is used by WLM to manage the work. Execution states refer to using and delay states which are continuously collected by WLM. The goal definition which is based on execution states is the execution velocity goal. Please refer to [7], [3], or [12] for more detailed explanations on WLM goals.

[1] This is only really true for CPU and I/O resources but for simplification we assume an enclave can be managed independently.

2. Management of CICS and IMS

2.1. Infrastructure

CICS with Multi Region Option (MRO)[1] and IMS create a work manager consumer model to process transactions or units of work. Figure 2.1 shows the general structure of such a model.

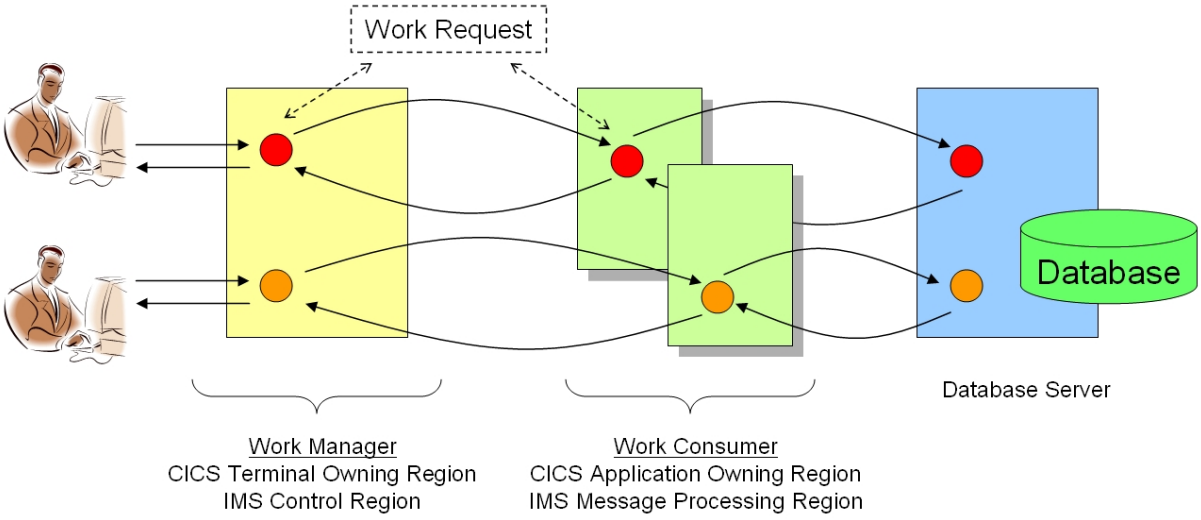


Figure 2.1.: Work Manager and Consumer Model

For this programming model a work manager receives work request from outside the z/OS system through a network connection or from another application running on z/OS. The work manager pre-processes the work request. One of these pre-processing steps is to classify the request to WLM. This classification passes attributes of the request to WLM and WLM compares the attributes with definitions for the subsystem created by the installation. As a result WLM associates the request with a user defined service class and returns a classification token to the subsystem work manager.

The work manager now initializes the work request and associates it with a construct which allows WLM to determine which work requests execute in the subsystem address spaces. This construct is named a performance block (PB). It keeps the classification information, the service class token and state information of the transaction. Then the work manager dispatches the transaction to a TCB and starts its execution. In order to process the request of the end user the subsystem needs to start one or multiple programs. This is usually not done in the work manager which receives the request. The request is passed to work processing or work consumer region. Figure 2.2 shows an example for a transaction flow. This example is for CICS and the region which processes the work request an Application Owning Region (AOR).

Figure 2.2 also depicts how the transactions are monitored in the CICS regions. Each transaction is associated with a performance block (PB) in each region. When the TOR passes the request to the AOR it marks it in its local performance block as being switched for processing to another region. Then it

[1] More processing models also more simplistic models are described in [13] Chapter 2.

dispatches another transaction. The PB remains associated with the work request and thus allows WLM to determine that the processing of the request is still ongoing.

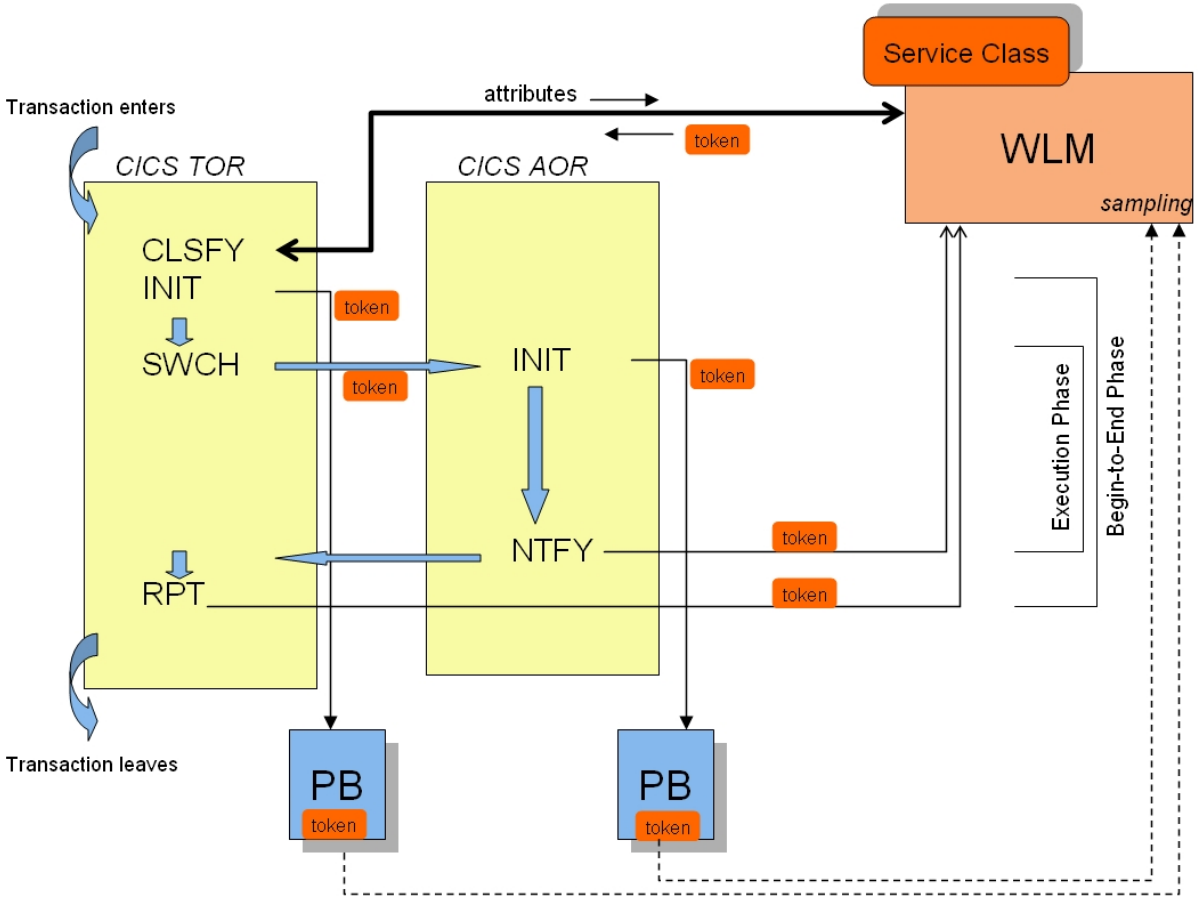


Figure 2.2.: CICS Transaction Flow and Tracking

When the AOR receives the work request it also associates a performance block with it to continue the book keeping for its processing. The AOR also dispatches the transaction and starts a program to process the end user request. During the processing it is possible that the request is passed to other regions and that the program accesses a database to receive and store data. These various processing steps are marked in the PB. When the request finally completes in the AOR its completion is notified (NTFY) to WLM and the work request is returned to the TOR. The notification saves the execution time of the request within the AOR and releases the local PB of the AOR from the transaction.

Finally the TOR will pick up the request again, potentially performs some post processing and ends the request by reporting (RPT) its end to end response time to WLM and also by releasing the local PB of the TOR from the transaction. From a WLM point of view the transaction is now completed and has ended.

WLM continuously monitors the performance blocks of the subsystem work managers. Every 250 ms it examines all PBs and collects information about which transaction is being processed by which subsystem address space and their execution states. The first information is used to associate the subsystem address spaces with the service classes being defined for the CICS and IMS transactions. The execution statistics are collected for monitoring reasons and passed to monitoring products like Resource Measurement Facility™ (RMF™).

A comprehensive description of the programming interfaces and programming models for subsystem work manager like CICS and IMS are described in more detail in [13].

2.2. Managing Subsystem Work Manager Address Spaces

Section 2.1 describes how a transaction is received by a work manager, how it is associated with a performance block and passed to other work manager regions for processing the end user request. The transaction is classified to WLM and and a PB is related to it in each work manager address space. At the end of each processing step WLM receives a notification of the execution respective response time of the work request and by continuously sampling the PBs it is able to determine which transactions are being processed by which regions. It should be noted that a region can process multiple transactions in parallel.

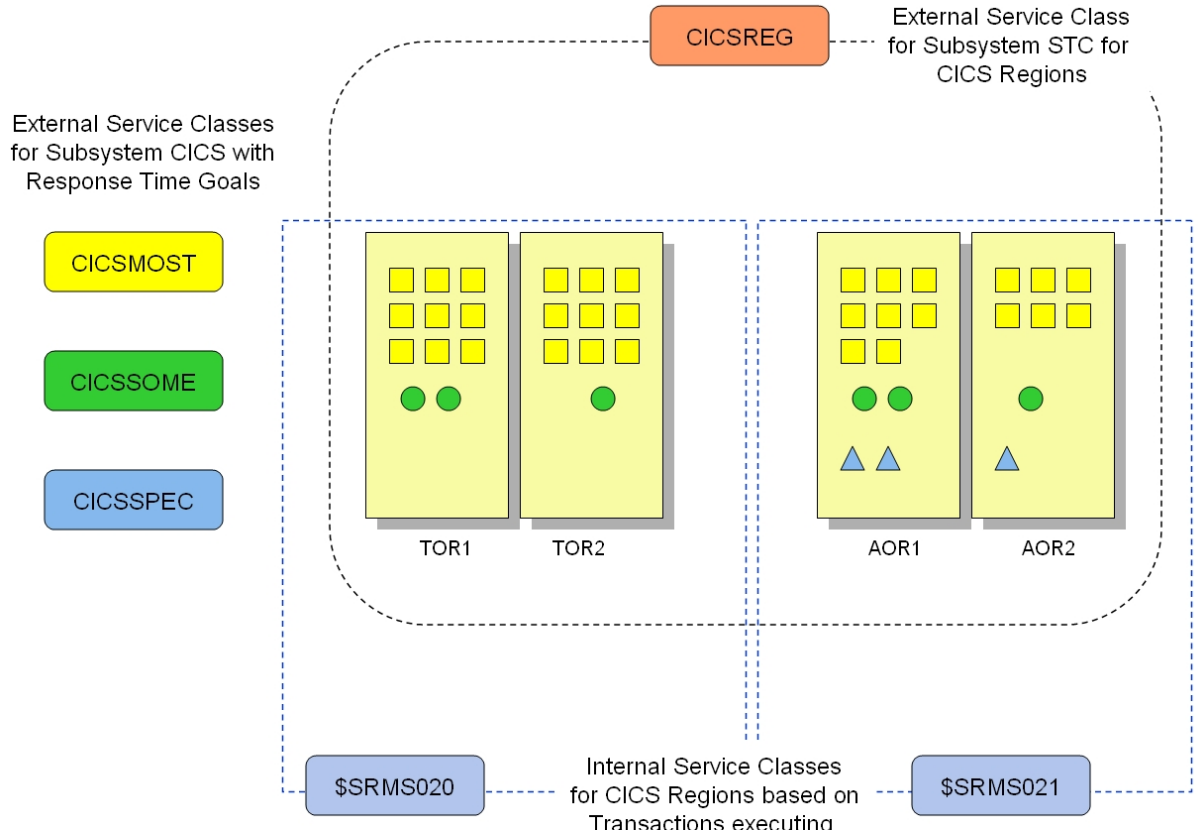


Figure 2.3.: CICS Server Topology

WLM uses the information of the response time returned at the end of the processing of the transaction to calculate the average response time of the CICS service class and to determine the response time distribution. This information allows WLM to determine the performance index for the service class (see [7]). Based on this information WLM is able to determine which service class needs help in case the performance goals are not met.

The information from the PB provides information for WLM to associate each region with a set of (transaction) service classes. A region can process transactions from different transaction service classes and therefore it is not possible to associate a region directly with a transaction service class.

Figure 2.3 shows a possible example for CICS regions (TOR1, TOR2, AOR1, AOR2, etc) processing

work for three transaction service classes (CICSMOST, CICSSOME, CICSSPEC). Based on the information which is continuously sampled from the performance blocks WLM is able to determine which transaction types are being processed by which region. Once WLM has derived a transaction type for a region from a performance block it keeps this association for at least 1 minute. This helps WLM to recognize also transactions which run very infrequently[2].

In this example WLM observes regions which process only transactions for service class CICSMOST and CICSSOME and regions which process transactions for all three external transaction service classes. It is not important how many transaction observations are being made for each region it is only important which mixture exists. Based on this mixture WLM creates internal service classes and moves the CICS regions from their external service class (CICSREG) to the internal class. In this example two internal classes exist:

- $SRMS020={TOR1, TOR2}: regions which process transactions for CICSMOST and CICSSOME
- $SRMS021={AOR1, AOR2}: regions which process transactions for all three external service classes

Note:
It is possible that AOR receive and return work requests without having a TOR involved. For CICS the characteristic of a region (whether its a TOR or AOR) is determined during runtime and potentially the region can perform any function.

2.3. Goal Assessment for Transaction Service Classes

WLM created a topology which associates the work manager regions with the external service classes for the transactions (see figure 2.3). This example results in a topology for service classes which connect the internal service classes for the CICS regions with the external service classes of the CICS transactions (see figure 2.4).

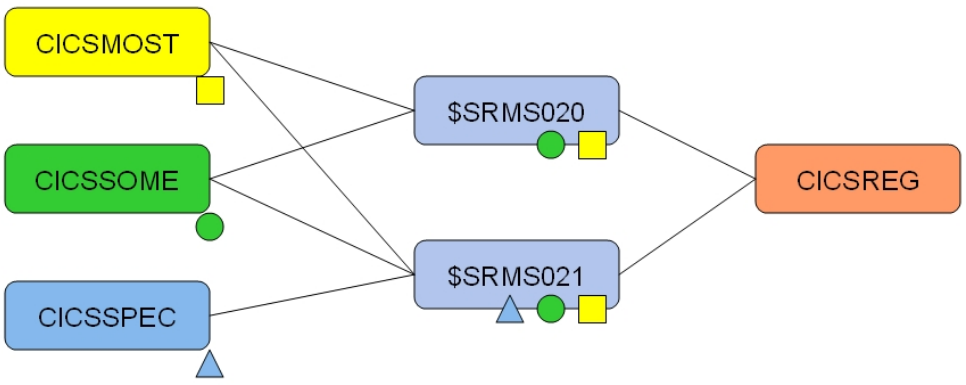


Figure 2.4.: CICS Service Class Topology

WLM determines the performance achievement based on the collected response time information for the external service classes. If a service classes does not meet its goals and needs help then WLM determines which internal service class contributes most to the external service class goal. In the current example external service class CICSSPEC is only associated with $SRMS021. So if CICSSPEC needs help WLM attempts to re-assign resources for the regions associated with $SRMS021.

For CICSMOST and CICSSOME the situation is more complicated because the regions processing these transactions are either associated with $SRMS020 or $SRMS021. WLM uses the summarized observations

[2]IMS Message Processing Regions typically only create 1 PB per region and process only 1 transaction at a time. WLM is able to associate different transaction types with a MPR because it keeps maintains the information of associated transactions for a period of time as described in the main text.

for the transactions for each region to determine which internal service class contributes most to the goal achievement of the internal service class. It selects an internal service class and then assesses whether it can help the internal service class. The overall goal assessment algorithm is described in [7], [3], and in [5]. [5] also describes the apportionment algorithm to identify the best internal service class in more detail.

This example shows that resources to meet the goals of a transaction service class are re-assigned on the basis of internal service classes. For this example it means if service class CIOCSMOST does not meet its goals WLM may select the internal service class $SRMS021 to change its dispatch priority. If that's successful the result is that WLM helped all transactions running in the regions associated with $SRMS021. So it helped indirectly also transactions for service class CICSSOME and CICSSPEC.

2.4. Subsystem Work Manager Topology

The topology in figure 2.4 is the result when each region processes transactions for nearly all external service classes. Only transactions for service class CICSSPEC are processed by a subset of the regions. As a result WLM creates two internal service classes for three external transaction service classes. But in general this is not the case.

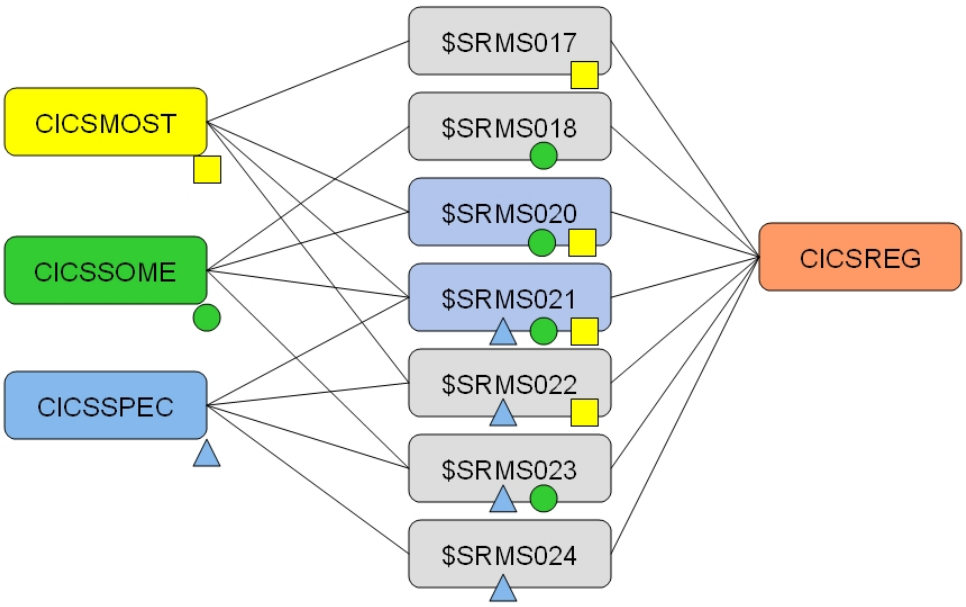


Figure 2.5.: General Subsystem Work Manager Topology

If enough regions exist it is much more likely to have many more internal service classes. Figure 2.5 shows 7 possible internal service classes for the 3 external service classes. This is the maximum possible number of internal service classes for 3 external service classes. It also assumes that at least 7 regions exist to process the transactions.

Note:
For n external transaction service classes it is possible that the number of internal service classes grows up to $2^n - 1$ assuming enough processing regions exist in the system.

This results in up to 3 internal classes for 2 external service classes, 7 for 3, 15 for 4, and already 31 possible internal service classes for 5 external service classes. As it will be shown later there is no advantage to have too many external service classes and at this point we can already conclude that the internal structure can become highly complicated if too many external classes exist.

2.5. Summary

WLM allows the subsystems CICS and IMS to classify their transactions to user defined service classes with response time goals. For managing the work WLM must also understand which subsystem region processes which type of transactions. It creates a topology which reflects the set of transaction types processed by the subsystem regions.

In case the goals for the transaction service classes are not achieved WLM uses the topology to identify the subset of regions (internal service class) which show the most effect on transaction execution. WLM then attempts to adjust the resource access for the selected internal service class in order to help the goal achievement for the external transaction service classes.

IBM Copyright © 2014

3. Defining Goals for CICS and IMS

Chapter 2 describes how WLM manages CICS and IMS workloads. The underlying assumption was that CICS and IMS work is managed by response time goals. We showed how the response time goals are being used to manage the CICS and IMS regions and emphasized the server topology which is created to recoignize which server regions processes which type of transaction (see section 2.4). At this point it is important to notice that management for CICS and IMS is always at the address space level and not for a set of individual transactions[1]. Therefore it is possible to manage CICS and IMS workloads either by execution velocity goals which have been defined for the CICS or IMS regions without defining goals for the CICS or IMS transactions. In this chapter we will explain why it is better to define goals for the transactions and manage the regions towards response time goals rather than execution velocity goals. We will also explain how it is easily possible to migrate an installation which currently uses execution velocity goals only to response time goals and we will explain what you avoid doing when you define service classes for CICS and IMS transactions.

3.1. Why using Response Time Goals

There are good reasons why you should use response time goals for CICS and IMS:

1. Response time goals provide a much better mechanism for monitoring and reporting CICS and IMS workloads.
   - It is very unclear how an execution velocity goal for an address space results in better or worse response times for the work running in the address space.
   - Using response time goals allows to define report classes for subsets of transactions. This allows very granular reporting and very detailed examination of work throughput and achievement.
2. A response time goal can be much more easily related to a Service Level Agreement (SLA). Without using response time goals it is always necessary to use additional information for example data contained in SMF 110 records for CICS or data from IMS logs to find out whether the throughput and transactions response times adhere to a service contract. Also it is necessary to use this information to find out how the selected execution velocity goals are related to the end user experience.
3. A response time goal is much more stable against configuration changes than an execution velocity goal.

In appendix A we will discuss in detail which events show through in response time goals and which in execution velocity goals and explain why a response time goal is better suited for transactional workload.

3.2. Using Response Time Goals

In the following we will define the steps to set up response time goals for CICS and IMS workloads. Our first approach is to set up goals for reporting purposes only. The assumption is that the CICS or IMS workload is managed towards execution velocity goals for the regions and no service classes exist for CICS or IMS transactions. In the first step we will set up an initial service class for the transactions and we will make sure that management of the workload does not change. In the same step we set up

[1] This is the case for enclave management

report classes to get a detailed view of different sets of online transactions. At that point we will achieve a comprehensive reporting infrastructure for CICS and IMS transactions.

3.2.1. Steps to create Report Classes for Online Transactions

In order to setup report classes for CICS and IMS transactions we will perform the following steps:

1. We define a single service class for the CICS or IMS regions. This should already exist because we assume that the CICS and IMS work is already managed by region goals. In case two service classes, one for CICS Terminal Owning Regions and one for CICS Application Owning Regions exist, and also one for IMS control region and one for IMS Message Processing Regions, we use this structure. Very often installations use one service class for all CICS regions but already a separation for IMS regions. Such structures are fine. In cases where the CICS or IMS regions are classified to a general STC service class with other address spaces we recommend to separate the regions from the other started tasks. In chapter 4 we will discuss cases where we recommend to define different service classes for CICS TORs and AORs but for the moment this is not urgent.
2. We assure that the management of CICS and IMS reamins unchanged. For this purpose we enter the WLM Administrative Application and select Option 6 "Classification Rules". Depending on whether our regions are started as Started Tasks or batch Jobs we select either the classification rules for subsystem STC or JES.On the classification panel we scroll two times to the right (F11) and change the column *Manage Region Using Goals Of* from TRANSACTION to REGION. Please make sure that you change this option for all region classifications.
3. Now we define a service class for CICS or IMS transactions. The service class must have one period and a response time goal. In case we have no idea about a good goal value we can estimate one and change it in a later iteration step.
4. The next step is to define report classes. For this step some knowledge about the transactions is required. We need an understanding which transaction belong together, whether it is advisable to classify transactions from where they come (for example the LUNAME), whether a transaction class is a good differentiation characteristic, or whether certain transactions should be identified by their names. In case we have no idea and in case we just want to perform some test we define only one report class. But it is absolutely necessary to define at least one report class because we will obtain transaction statistics in this step only from the report class.
5. Now we classify the transaction to the service and report classes. For this case we use Option 6 "Classification Rules" and enter the classification rules, service and report classes for subsystem CICS or IMS. We should at least enter the service class as default service class and always enter one report class as default report class.
6. This is all. We can now save the service definition, install it on the WLM Couple dataset, and activate it.

Note:
So far we haven't changed any management. If all steps have been performed correctly the CICS or IMS regions are still managed towards execution velocity goals. Step 2 is the most important step at this point.

3.2.2. Example

In this section we will illustrate the two important steps number 2 and 5 from the list above by using a simple scenario. For this scenario we have a simple CICS environment which executes only 3 transactions and which uses up to 5 CICS address spaces named CICSMON1 to CICSMON5. When we assume that a service class for the CICS regions already exists we have to make sure that the management remains towards the goals of the CICS region service class. Our CICS regions are started as batch jobs therefore the classification of the regions is under subsystem JES in the WLM Administrative Application. In case

that the regions are started as Started Tasks the modification to exempt the regions from transaction management must be performed under subsystem STC.

We start the WLM Administrative Application and select option **6.** "Classification Rules". On the subsystem type selection list we select subsystem JES with line command **3** "Modify". On the "Modify Rules for the Subsystem Type" panel we scroll twice to right using key **F11**. The we see the panel in the form as depicted in figure 3.1.

```
Subsystem-Type  Xref  Notes  Options  Help
-------------------------------------------------------------------------------
                 Modify Rules for the Subsystem Type     Row 1 to 12 of 12
Command ===> _____________________________________________ Scroll ===> PAGE

Subsystem Type . : JES         Fold qualifier names?   Y  (Y or N)
Description  . . . Batch Work

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                       <=== More
          --------Qualifier--------           Storage    Manage Region
Action    Type       Name     Start           Critical   Using Goals Of

 ____  1  TN         BVAU*    ___             NO         TRANSACTION
 ____  1  TN         CICSMON1 ___             NO         REGION
 ____  1  TN         CICSMON2 ___             NO         REGION
 ____  1  TN         CICSMON3 ___             NO         REGION
 ____  1  TN         CICSMON4 ___             NO         REGION
 ____  1  TN         CICSMON5 ___             NO         REGION
 ____  1  TN         CICSREG2 ___             NO         TRANSACTION
 ____  1  TN         CICS*    ___             NO         TRANSACTION
 ____  1  TN         QM*      ___             NO         TRANSACTION
****************************** BOTTOM OF DATA ******************************
```

Figure 3.1.: Define manage regions using the goals of the regions

We change the column "Manage Regions Using Goals Of" from TRANSACTION to REGION for our CICS regions (CICSMON1 to CICSMON5). This assures that the management of the CICS environment which is related to this regions is exempted from transaction management after the WLM service definition has been installed on the WLM couple data set and a service policy has been activated. In chapter 4 we will discuss the possibilities to exempt regions from management in more detail.

The next step is to define a service class and one or multiple report classes for CICS. This can be done by using option **4.** (Service Classes) and **7.** (Report Classes) from the "Definition Menu" of the WLM Administrative Application. The service class is required but the goal doesn't matter much at this point. If you have no idea which goal would apply to your workload we simply suggest to use a goal of 80% < 0.5 sec as a starting point.

The report classes are more important for the moment. For them you should have an idea what type of transactions execute, which transaction classes are being used, or from which LUNAME the transactions come into the system. The available classification rules are described in [11] and specifically for CICS [16] and [17], and for IMS [18]. Again in case you simply start to evaluate the possibility how to perform the migration step and you must at least define one report class. This is important because as long as the regions are exempted from transaction management the transaction data is reported only through the report classes.

In our case we use three different transactions CIC1, CIC2, and CIC3 and we assign three report classes, one for each transaction as shown in figure 3.2. In most cases it is not meaningful to classify each transaction individually. It is much more meaningful to use either transaction classes or LUNAME as classification mechanism. The classification for individual transactions in larger environments makes only sense if you have a need to monitor the performance of some very specifically.

```
Subsystem-Type  Xref  Notes  Options  Help
------------------------------------------------------------------------------
                Modify Rules for the Subsystem Type       Row 1 to 3 of 3
Command ===> _____________________________________________ Scroll ===> PAGE

Subsystem Type . : CICS        Fold qualifier names?   Y  (Y or N)
Description  . . . CICS Classification Rules

Action codes:   A=After     C=Copy        M=Move     I=Insert rule
                B=Before    D=Delete row  R=Repeat   IS=Insert Sub-rule
                                                        More ===>
          --------Qualifier--------                -------Class--------
Action    Type       Name     Start                 Service     Report
                                              DEFAULTS: CICSMED     RCICSDEF
 ____  1  TN         CIC1     ___                   CICSHIG     RCICSA00
 ____  1  TN         CIC2     ___                   CICSHIG     RCICSA99
 ____  1  TN         CIC3     ___                   CICSHIG     RCICSA33
****************************** BOTTOM OF DATA ******************************
```

Figure 3.2.: Classify CICS (IMS) transactions to service and report classes

The next step is to save the WLM service definition, install it on the WLM Couple data set, and to activate one of its service policies. After performing these steps we can immediately monitor the results of our changes assuming we have work which is executed for our CICS or IMS environment. For this reason we start RMF Monitor III and select the System Summary report (SYSSUM) which gives us the best overview for goal achievement. Figure 3.3 shows an example of the System Summary report after we activated the changes discussed in the previous steps.

The report in figure 3.3 shows our CICS service class CICSHIG with the goal definition of 80% < 0.5 sec. We also notice that the "Trans Ended Rate" and the response time columns show only zeros for the service class. The activity data is reported for the report classes. Our three report classes RCICSA00, RCICSA33 and RCICSA99 show transaction ended rates, execution and actual response times. The wait time is zero because there is no internal queuing and typically this column should remain zero for CICS environments.

In figure 3.4 we show an excerpt from the same environment after we have changed the regions CICSMON1 to CICSMON5 from REGION to TRANSACTION management (see figure 3.1). The snapshot in figure 3.4 now shows that the transaction data is reported for both the service and the report classes. because we haven't changed the load the numbers reported in both examples are about the same and the the sum of the report class data is reported in our example for the service class.

For a real migration we recommend that you first start to layout your reporting structure, define one service class, exempt the regions from transaction management as described before and then monitor the system for a while. During this monitoring it is important to understand the performance of your CICS or IMS environment and define a goal which is meaningful when you switch to transaction management.

At this point it is also necessary to take a look at a pragmatic approach for the migration. We always recommend to specify report classes for detail analysis. We definitely recommend to keep the number of

different service classes small, to 1 or 2, and use report classes as many as you like.

```
HARDCOPY  RMF V1R12  Sysplex Summary - WLM1PLEX        Line 1 of 13
Command ===>                                            Scroll ===> CSR
WLM Samples: 240     Systems: 1  Date: 12/14/13 Time: 15.57.00 Range: 60    Sec

Service Definition: zosdemo               Installed at: 12/14/13, 15.54.50
     Active Policy: ZOSDEMO               Activated at: 12/14/13, 15.54.57

               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time

STC      W          0.0                              0.000 0.000  0.000  0.000
STCDEF   S  3    30 0.0                           N/A  0.000 0.000  0.000  0.000
SYSTEM   W          100                              0.000 0.000  0.000  0.000
SYSSTC   S      N/A 100    N/A                       0.000 0.000  0.000  0.000
UNIKABTC W           95                              0.000 0.000  0.000  0.000
BTCHCRIT S  1    50  95                          0.53  0.000
ZOSDEMO  W           95                              0.000 0.000  0.000  0.000
CICSHIG  S  2        95  0.500 80%         N/A   N/A  0.000 0.000  0.000  0.000
RCICSA00 R          N/A                              0.500 0.000  0.561  0.561
RCICSA33 R          N/A                              0.633 0.000  0.588  0.588
RCICSA99 R          N/A                              0.117 0.000  0.904  0.904
RDEFBTCH R           95                              0.000 0.000  0.000  0.000
RDEFSTC  R          4.8                              0.000 0.000  0.000  0.000
```

Figure 3.3.: RMF SYSSUM report for region management

```
               ------- Goals versus Actuals --------  Trans --Avg. Resp. Time-
               Exec Vel  --- Response Time ---  Perf  Ended  WAIT EXECUT ACTUAL
Name     T  I  Goal Act  ---Goal--- --Actual--  Indx  Rate   Time   Time   Time
...
CICSHIG  S  2        96  0.500 80%         54%  1.50  1.200 0.000  0.644  0.644
...
RCICSA00 R           96                              0.483 0.000  0.632  0.632
RCICSA33 R           96                              0.617 0.000  0.634  0.634
RCICSA99 R          N/A                              0.100 0.000  0.761  0.761
...
```

Figure 3.4.: RMF SYSSUM report for transaction management

For migration it might be better to use only a few report classes unless you have a reporting mechanism which allows you to easily derive a service class goal from multiple report classes because there is no summary available for all report classes. This is the case once you have switched to transaction management[2].

It is probably necessary to adjust the goal. From the report classes you can obtain the response time

[2]The service class is summary across all of its report classes

distribution as shown in figure 3.5. A meaningful goal can be derived from the weighted response time across all report classes by using the transaction ended rate as weighting factor. For the percentile we recommend to use the summed buckets for all response times which are also available for the report classes. If you create a weighted average for the sixth bucket which contains the sum of all ended transactions of bucket 1 to 6 then you have a very good approximation of the achieved percentage of your goal definition. Another approach is to use only very few perhaps only one report classes during migration and extend the number once you have switched to transaction management.

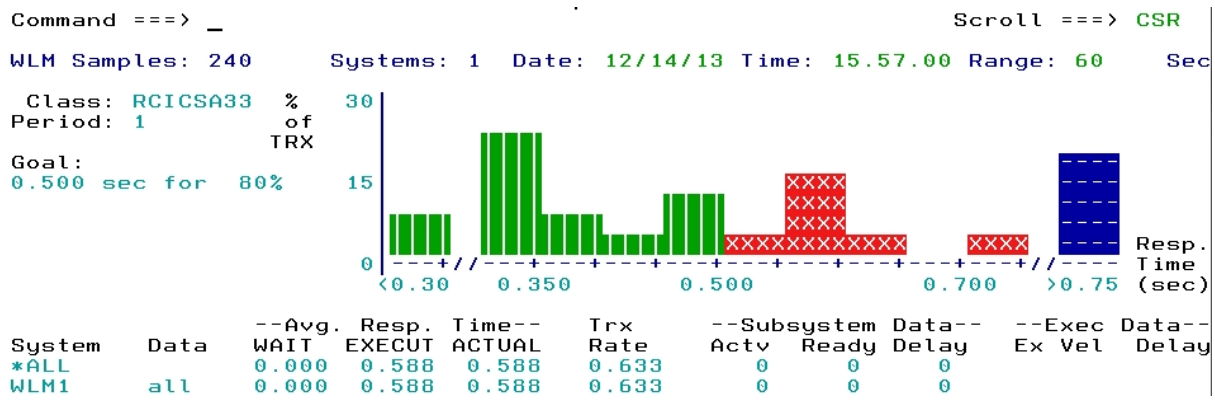


Figure 3.5.: RMF Monitor III Response Time Distribution report (SYSRTD)

3.3. Migration to Response Time Management

The migration to response time management for CICS is pretty easy when you follow the steps described in section 3.2. Before you complete the migration you should run your environment with report class definitions for your transactions and exempt the regions from transaction management in order to observe the behavior of your environment and to derive a meaningful goal. As pointed out in section 3.2.2 it might be the best to start very simple with one service class and only one report class. The number of report classes can be extended at any point in time with no impact to the management of the transaction environment. An interesting question is how many service classes you should use and why we recommend very few service classes?

One important aspect is related to the way how WLM manages CICS and IMS environments. We discussed the internal structure of CICS and IMS management in sections 2.2 and 2.4. WLM determines the types of transactions which are executed by the work manager regions and then groups regions which process the same type of transactions into internal service classes. As shown in figure 2.5 it is possible that a relative high number of internal service classes is created for few external service classes. The question is whether this is a problem? The effect of having multiple internal service classes which process very similar sets of transactions results potentially in different service for the same type of transaction. In our example most of the transactions execute in service class CICSMOST. In figure 2.5 CICSMOST transactions were spread across 4 different internal service classes. In cases when CICS transaction goals are not met WLM tries to help the external service class which doesn't meet its goals. In order to do this WLM selects one or multiple internal service classes which contribute to the goal achievement of the external service class. The important part is that it is very likely that not all internal service classes will be selected to help an external service class. As result the access to resources becomes different for different internal service classes. When we assume that CICSMOST is the transactional work with the highest throughput and that it is spread across 4 different internal service classes it is very likely that CICSMOST transactions get different access to system resources depending on the internal service class

they run in. In appendix B we will discuss an example that consists of multiple external service classes which result in different management for the same transaction types.

Note:
Multiple external service classes for CICS and IMS can result in different management for the same type of CICS and IMS transactions. As a result the external response time of such transactions depends of the chance which CICS or IMS regions process them and which other transaction type execute at the same time.

Another aspect to consider is the type of response time goal which you should use for managing your transaction environment. The best solution is to use a percentile response time goal. This goal focuses on the transactions which end within the goal or which just run a little longer. WLM always attempts to increase the number of transactions running within the goal definition in cases goal definitions can't be met[3]. When we look at figure 3.5 we see the response time distribution of one of the report classes. Let us assume this is a service class. the goal definition was 80% of the transactions should end within 0.5 seconds. We can see that this is not achieved and perhaps our goal definition was too optimistic. When we assume that around 20% of the transactions need an execution time of more than 0.7 seconds we could change our goal to 80% < 0.7 seconds. Now the graphic would change and we will get a much more realistic picture. WLM will now have a chance to meet the goal but the important aspect is that the long running transactions are no problem. They get the same service as the majority of other transactions and they do not disturb the management. In case we would define a separate service class for them and they execute in the same regions than the other transaction they will generate a problem and disturb a uniform transaction management. So it is much better to leave them in the same service class and just define one or multiple report classes to monitor them.

Note:
Percentile response time goals are very stable against few outliers. Once a percentile goal has been adjusted to a realistic definition some long running transactions do not disturb the management and can share service classes with short running transactions.

Both scenarios show that WLM will manage transaction environments towards the service classes which contain the highest transaction ended rates. Defining service classes for some few long running transactions is not helpful for the main set of transactions and most often results in different external behaviors. On the contrary the long running transactions will get a free ride even if they are defined to service classes with low goals. Therefore it is better not to separate them to different service classes as long as they share the same regions.

Note:
Transactions defined to service classes with low goals will often get the same treatment as transactions with very stringent goal definitions in case they share the same regions for processing.

This discussion also answers the question when it is useful to define separate service classes for transactions. It really only makes sense if transactions being defined to different service classes also execute in disjunct sets of regions. In this case WLM can provide the treatment to the transactions which is really related to their service definition.

Note:
Different service classes for different types of transactions is only meaningful if the transactions execute in different sets of regions.

[3]Assuming this is at all possible

IBM Copyright © 2014

4. Advanced Management Options

Chapter 2 gives an introduction how WLM manages CICS and IMS workloads using response time goals and chapter 3 describes the step how you can migrate to response time management for your on-line transaction environment. In this chapter we will take a look at some advanced options for managing CICS and IMS workloads such as exempting regions from transaction management, using the new option BOTH, instead of completely exempting the regions, and at what point you could use CPU and Storage Critical options.

4.1. Exempting Regions from Transaction Management

In chapter 3 we described a methodology to migrate from execution velocity to response time management for your CICS and IMS workloads. During this migration we exempted the regions from being managed towards response time goals because we wanted to monitor the system first and derive a realistic goal for response time management. Figure 3.1 depicts a WLM Classification menu on which we changed the default from TRANSACTION to REGION for our CICS regions.

Note:
A valid scenario for exempting regions from response time management is during migration of the management options for your transaction environment.

Another valid scenario exists when you run production and test environments either on the same system or within the same sysplex. For CICS or IMS test environments it is very often counter productive to use response time management because there is usually to little traffic running through these environments. Appendix A compares execution velocity and response time management. Response time management reflects much better the behavior of the transaction environment rather than configuration options of the system. But it also requires a certain amount of transactions being processed. For response time management WLM collects data from performance blocks every 250 milliseconds. This is not required for an environment which runs more sporadically without a specific goal which needs to be related to an end user service level agreement.

Note:
A valid scenario for exempting regions from response time management is during migration of the management options for your transaction environment.

Finally it is possible to exempt back end regions from response time management. This means you should never specify REGION management for TORs when you plan to use TRANSACTION management for your AORs because the end-to-end view must be maintained for management. On the other hand it is possible to exempt an FOR[1] or AORs which function as back-end regions in the transaction environment. Whether a good scenario exists for which this is a valid option depends on the specifics of the installation.

Note:
It is important that you either exempt all regions of a subsystem from being managed towards TRANSACTION goals or the backend regions only. You should never exempt CICS Terminal Owning Regions but leave the CICS Application Owning Regions being managed towards transaction goals.

[1] File Owning Region

4.2. Combining Response Time and Execution Velocity Management

A new option for managing CICS and IMS transaction environments has been introduced with APAR OA35248 on z/OS V1.10 and all higher z/OS releases. The new option allows to exempt the front-end regions and maintains the end-to-end context of the CICS and IMS transaction flow.

CICS as well as WebSphere or DB2/DDF adheres to a Work Manager/Consumer model. That means some regions (TORs) are work receiver, sender of the results to the work originator and distribute the work to consumer regions (AORs) which start application programs to perform the functions behind the work requests. The work managers typically only require very short access to resources but they also need very fast access to the resources in order to avoid being a bottleneck. The work consumers typically run more resource intensive programs which do not require the same fast access to resources.

For WebSphere and DB2/DDF the model is supported in a way that the work managers run as server processes in a service class with a high importance and high execution velocity goal. They do not consume a lot of resources but the service class definition allows them to get fast access to resources if necessary. The work is encapsulated in enclaves and the server region tasks join these enclaves. The enclaves are typically classified to service classes with lower importance and lower goals than the service classes for the work manager regions.

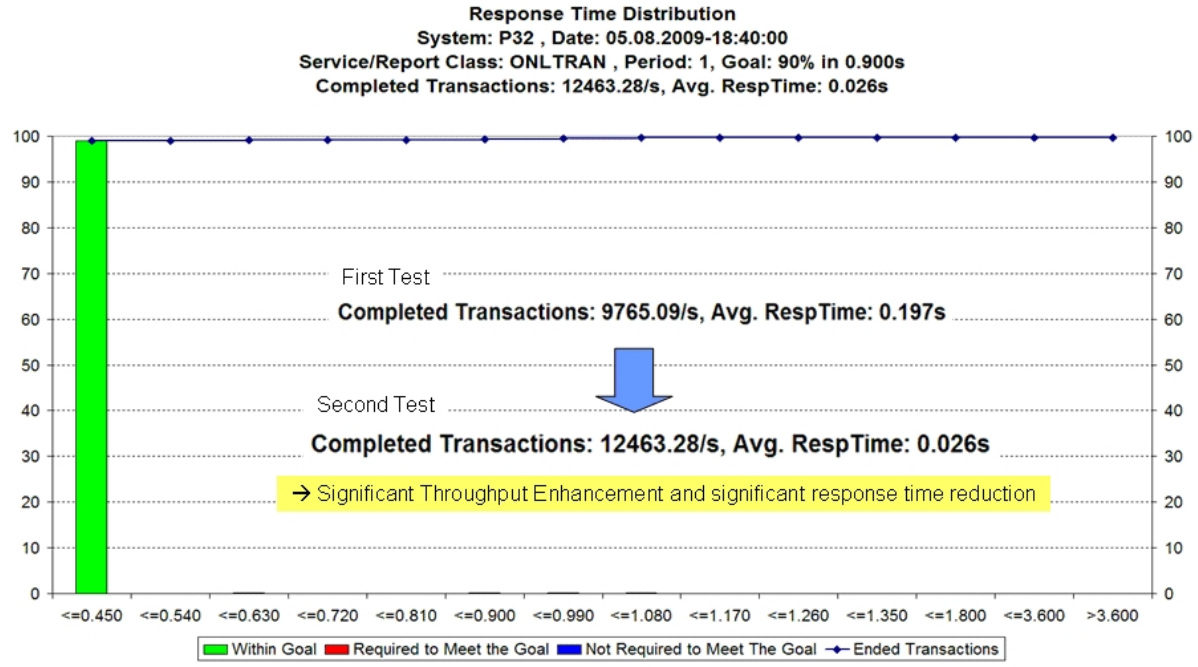


Figure 4.1.: Test result for using option BOTH in a CICS environment

For CICS and IMS this model was not supported yet. All regions are treated the same if the work is managed towards transaction goals and distinction is just made from the mix of transactions running in them. Nevertheless for CICS the same model exists as for WebSphere and DB2/DDF. CICS TORs typically function as work receiver and sender of the processing results. The TORs classify the work requests, perform some administrative tasks and then distribute the requests to CICS AORs which execute the application programs. During LSPR tests as well as in customer environments which run exclusively CICS a negative effect has been recognized if all regions are treated the same.

Typically AORs consume much more CPU than TORs. When contention occurs it is now possible that TORs do not get fast enough access to CPU because they have to wait until the AORs complete their time slices. As a result at higher utilization levels, typically above 85% a noticeable queue delay within

the CICS TORs can be recognized. This reduces the end-to-end response times of the CICS transactions and the throughput of the CICS work.

Figure 4.1 depicts the results of a LSPR[2] test scenario for a CICS environment. The workload consisted of a Websphere Application Server which receive requests and send them to a set of CICS TORs. The TORs classify the work and send the requests to AORs which start application programs and access a DB/2 data base. Once the workload increases, significant contention and delay was recognized within the CICS TORs. The solution is to exempt the TORs by maintaining the end-to-edn context of the CICS environment. This is accomplished by using option BOTH on the WLM Classification menu (see figure 4.2). The TORs will now be managed towards the execution velocity goals of their STC or JES region service class and the AORs are still managed towards the CICS response time goals.

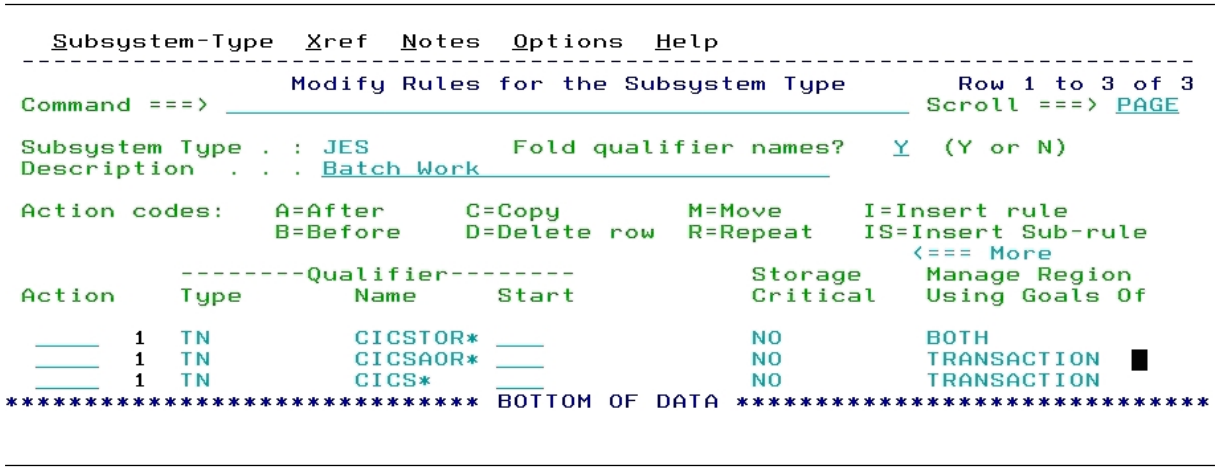


Figure 4.2.: Defining option BOTH for CICS TORs

A few remarks are necessary with respect to option BOTH:

- Typically TORs which just receive and send requests from and to the network consume very little compared to AORs which start application programs for work requests. The consumption is typically in the range of 5 to 10% of the AORs.
- If you want to use the option you should classify the TORs to a separate region service class and give this service class a higher importance and a stringent execution velocity goal.
- TORs and AORs can always function in the same way. That means both regions can receive requests and process them. Therefore it is advised to only separate regions which function as sender and receiver of work (TORs).
- For IMS in theory the same applies but we haven't noticed this phenomenon for IMS workload yet.
- Using option BOTH is recommended when you use an environment which is dominated by CICS workload and on which other work like batch is either not present or stalls during peak periods.

4.3. Using CPU and Storage Critical Options

The question whether to use CPU and Storage critical is of general nature. CPU Critical protects the Dispatch Priority of service classes against lower important work. Storage Critical does the same for the working set of service classes during periods of low activity. A description of the functionality of both options can be found in [7].

[2]Large System Performance Reference

Storage critical makes most sense on systems with alternating Batch and transactional workload and a configuration which makes it possible that the occupied main memory of the transactional workloads is paged out during Batch runtime. This highly depends on the amount of main memory installed on the z/OS system. For most modern systems the main memory should probably be sufficient to maintain the transactional workload during off shift but Storage protection is a viable option where this isn't the case. A simple indicator exists when paging occurs at the time when the on-line workloads becomes active again at start of prime shift. A viable alternative to Storage Critical is to install more memory.

Note:
Storage Critical is a viable option on production systems with small amounts of storage. For most large production environments it is probably not necessary anymore.

CPU Critical is a little different. The protection is active when work executes. Thus the necessity for Storage Critical highly depends on the goal definitions and goal achievement of the service classes. This also means it depends on the goal type which is used for the work. In appendix A we discussed the difference between execution velocity and response time goals. For a response time goal it is pretty easy to define a goal that ensures an automatic protection of the work (see figure A.2 and A.3). In this example the goal is defined in a way that the work is always protected. For execution velocity goals this is more difficult. Especially on systems with many processors execution velocities of high important service classes have a tendency to very high values. In certain cases it might be necessary to define very high execution velocity values. For the example in appendix A an execution velocity goal which would be equivalent to the response time goal would be higher than 80 (see figure A.4). Because execution velocity goals depend highly on configuration of the system it requires constant maintenance. This is very often the reason why installations choose the CPU Critical option for their most important transactional workload if they use execution velocity goals to manage them.

Note:
A response time goal provides the option to avoid CPU critical because the goal can easily be tailored to a value which meets the installation expectation.

In section 4.2 we discussed the possibility to define the TORs to a separate service class, define an importance level to this service class which is above the importance of the CICS transactions and to use option BOTH to manage the TORs towards an execution velocity goal. The reason is that we want to give the regions which receive and send work from and to the network preferential access to CPU. For this case we also advise to set CPU Critical for this service class. It typically consumes little CPU compared to the AORs and its Dispatch Priority should also be higher than the Dispatch Priorities of the transaction service classes.

Note:
CPU Critical makes most sense for high important work which needs fast access to CPU and which doesn't consume high amounts of CPU. Work manager control regions as well as CICS TORs are good examples.

IBM Copyright © 2014

5. Summary

The idea of this paper is to summarize the workload management aspects for CICS and IMS work on z/OS. Especially with chapter 3 we want to motivate the use of response time goals to manage these transaction environments. The most important values for response time goals are that the results can be easily correlated to the end user experience and that the goal definition is more stable against configuration changes than execution velocity goals. We also tried to show that with little effort and no danger it is possible to migrate an environment to response time goals.

You should get some additional information from this paper in addition to the documents listed in the Literature section. It is highly recommended that you read the WLM redbook (see [12] and the WLM Planning manual [11] before you start modifying your service definitions. For further reading the documents [2] and [3] are recommended as well as the WLM redbook [12]. Additional documentation can be found on the internet at

WLM http://www.ibm.com/servers/eserver/zseries/zos/wlm/

IRD http://www.ibm.com/servers/eserver/zseries/ird[1]

SWPRICE http://www.ibm.com/servers/eserver/zseries/swprice/[2]

WSC http://www.ibm.com/support/techdocs[3]

One important part of revisiting your goal definitions is the necessity to measure the goal achievement of your work and to analyze the workload behavior. If you do not have a tool for this purpose it is recommended to take a look at

RMF http://www.ibm.com/servers/eserver/zseries/zos/rmf/

and download the RMF Spreadsheet Reporter. It helps you to do trend and workload analysis based on RMF SMF data.

The focus of this article is the response time management for transactional workloads. But there is much more, like Intelligent Resource Director which expands WLM capabilities to the LPARs running on the same Central processing Complex, Parallel Access Volumes which were briefly mentioned and which dramatically improve the I/O performance of a z/OS system, and Batch Management which underwent a set of improvements with z/OS 1.4 and gives you the ability to manage your batch work efficiently in a sysplex environment.

[1] Intelligent Resource Director
[2] Software Pricing
[3] Washington Systems Center

A. Comparison of Response Time and Execution Velocity

We will take a look at the behavior of response time versus execution velocity. To compare both metrics we take a look at a CICS environment and compare the execution velocity of the CICS regions with the transaction response times of the CICS transactions. It should be noted that CICS workload is managed towards response time goals in this case. nevertheless the comparison is legitimate because we will emphasize the events which show through in the execution velocity goals and which show through in the response time goals.

A.1. Scenario

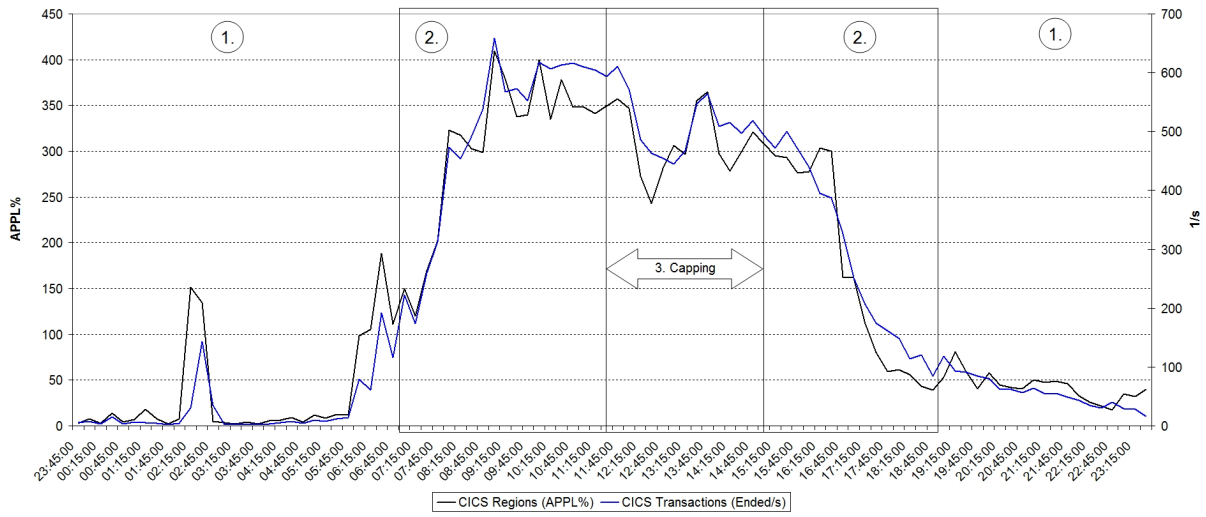


Figure A.1.: Application Utilization for CICS regions and transaction rate for CICS transactions

Figure A.1 shows the workload utilization and ended transaction rate of a CICS environment running under z/OS. The markers denote the different processing time periods of the day:

1. This is the night period with only very little to no transaction activity.
2. During prime shift which starts around seven in the morning and which lasts until seven in the evening the processing increases up to 600 transactions per second. The CPU utilization of the CICS regions grows up to 4 logical processors[1]
3. Between 12am and 3pm the system is capped by a group limit[2].

[1]The CPU utilization of the CICS regions is measured in Application Utilization (APPL%) which is an RMF metric. This is the converted application time into an overall percent of a single CPU used for the workload. A value of 100 means that the workload consumes CPU of one logical processor.

[2]The LPAR belongs to a capping group therefore the capping is not uniform and depends also on the consumption of other partitions in the same group

We will now analyze the goal of the workload its performance achievement and the execution velocity for the regions for these segments.

A.2. Response Time Analysis for CICS

The CICS transactions are classified to a single service class in WLM. The goal definition for the service class is via a percentile response time goal which requires that 92% of all transactions end within 0.4 seconds. The average response time and the performance index are depicted in figure A.2. When we analyze the segments we can observe:

1. During off shift with very small numbers of ending transactions (see also figure A.1) both the average response time and the goal achievement show a very wide range.
2. During prime shift with at least some ending transactions up to high numbers of ending transactions the average response time becomes very stable. The performance index now ranges between 1 and 1.2[3].
3. Even for the capping period the response time does not really change and the performance index shows the same range as for the other prime time periods.
4. We recognize another single event at the beginning of the capping period. At this point we can't tell where it comes from and in fact this event has a different cause[4]. It is not related to the capping event and occurs after capping was already active.

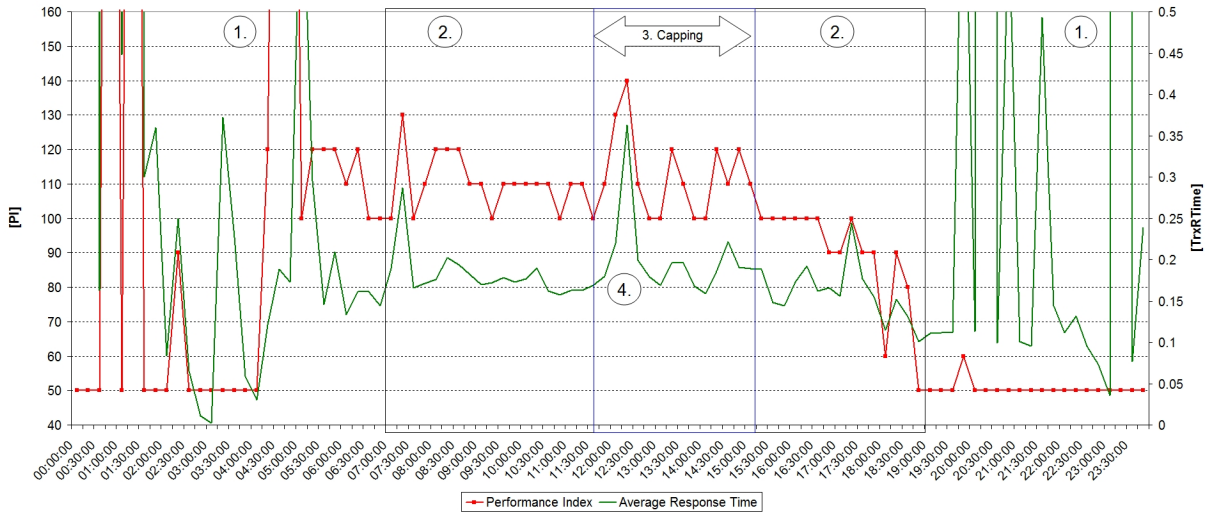


Figure A.2.: Average Response Time and Performance Index for CICS transactions

At this point we can state that a response time metric is very depending on the number of running and ending transactions for a workload. It seems very stable to LPAR wide events like capping and it reflects other potentially workload related events as the peak in bullet 4 shows.

The response time goal is not defined as an average response time. Therefore it also seems necessary to take a look at the number of ending transactions. Figure A.3 shows the summarized percentages of ending transactions within the goal value of 0.4 seconds (blue curve). We can observe that the curve is always slightly below the 92% value of the goal definition during prime shift. This results in the performance index between 1 and 1.2 as shown in figure A.2. During off shift the curve also shows a wide range but not

[3]The PI is scaled by 100 in figure A.2
[4]We will only analyze the event in the way how it can be recognized by the different metrics and not its root cause.

so drastically than the average response time. The event marked as **4.** is shown as a slight dent which tells us that some transactions required a longer execution time during this period. The graphic also shows the corresponding curves for the number of ending transactions within half and twice the goal value and we observe that they frame the goal percentage curve.

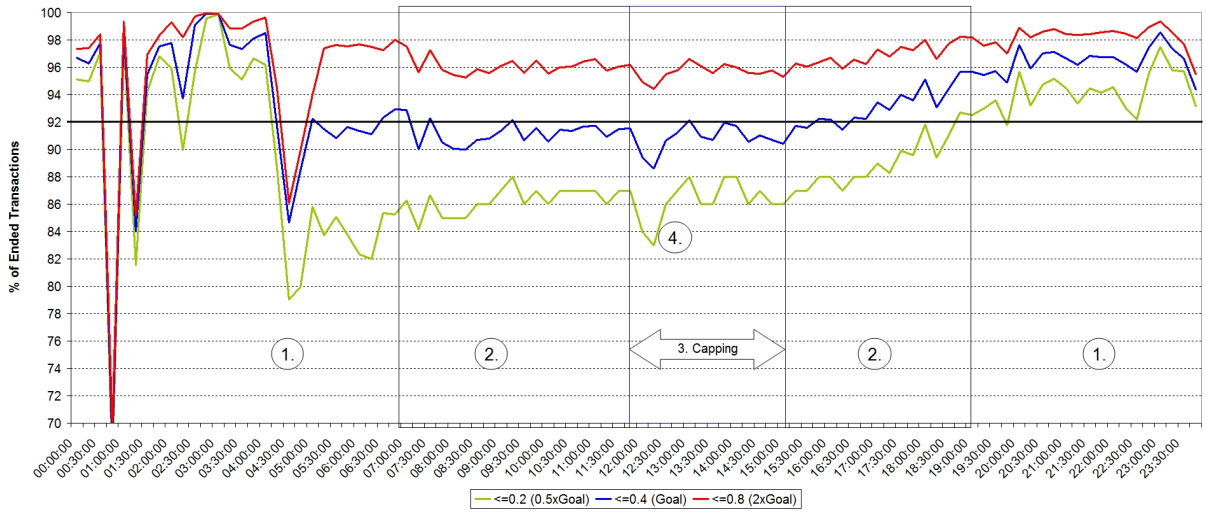


Figure A.3.: Response Time Percentiles

A.3. Region Analysis

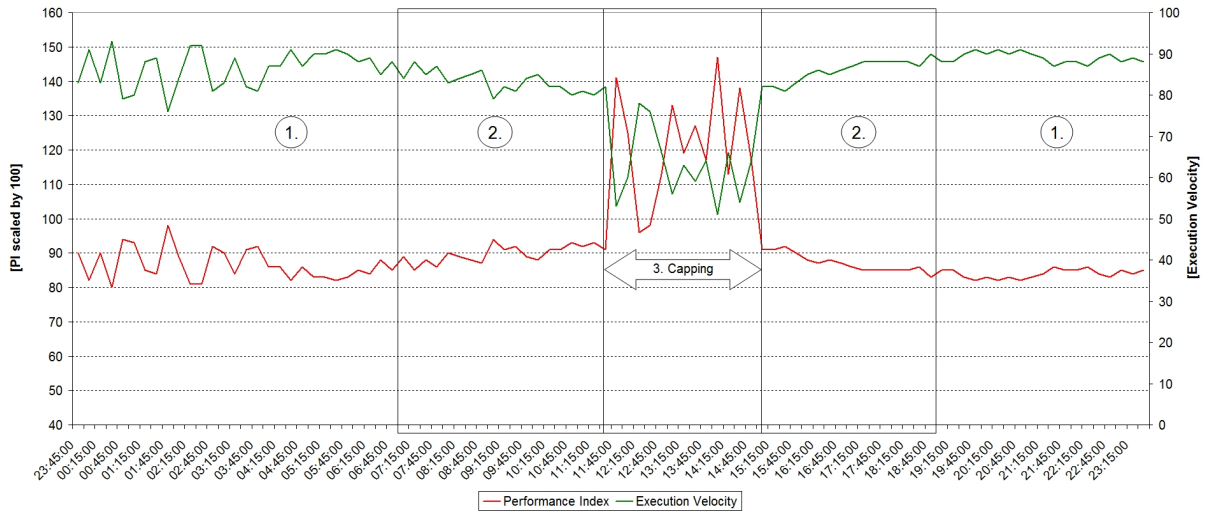


Figure A.4.: CICS Regions: Performance Index and Execution Velocity

Figure A.4 depicts the performance index and execution velocity for the service class of the CICS regions. The application utilization of the regions was already shown in figure **??**. The performance index for the regions is not used by WLM for management purposes because the regions are managed by the goals of the CICS transactions. Nevertheless it is calculated and reported. By analyzing the graphic we can conclude:

1. The execution velocity is very stable between 80 and 90 during off shift. This is the first remarkable difference compared to the transaction response time.
2. During prime shift on an unchanged system the execution velocity is again stable between 80 and 90.
3. This changes drastically during capping and we will discuss why this happens.
4. We cannot really identify this event on this chart. In fact it seems that this event is represented by a dent going back to an execution velocity and performance index as we observed it before. We will later discuss this in more detail.

The interesting question is why the execution velocity starts to become worse during the capping period. The reason is simply that the logical configuration changes during a capping event. The system runs on a zEC12 with Hiperdispatch turned on. Without capping the system has 9 vertical high and 1 vertical medium processor defined. During capping this value is reduced down to 6 vertical high processors. Also during capping the low processors are being parked. This results in a much more stringent configuration than before. We did not observe any negative impact on the transaction response time but the execution velocity seems to be sensitive on system configuration. This is indeed the cases even worse. Every change in system configuration shows an impact to the execution velocity. Every migration to a newer system generation with potentially more or less configured logical processors shows an impact. As a result an execution velocity goal needs constant maintenance while a response time goal is much more insensitive against changes to the system.

Note:
Execution Velocities and execution velocity goals are very sensitive to changes in system configuration.

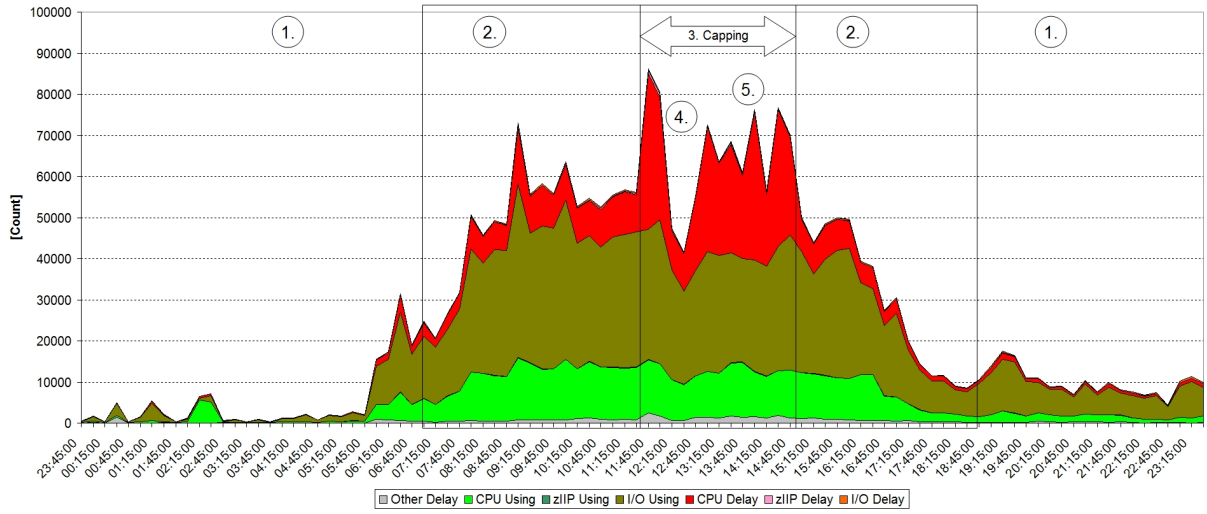


Figure A.5.: CICS Regions: Using and Delay samples

The execution velocity is calculated by dividing the using samples of a service class with the total productive samples:

$$Execution\ Velocity = \frac{All\ Using\ Samples}{All\ Using\ +\ AllDelay\ Samples} \times 100$$

Figure A.5 shows the using and delay samples collected for the CICS regions. The interesting period is again the capping time. First we can observe again event number **4** which is shown as a very visible reduction in CPU delays. When we take a look back at figure A.1 we can also observe that the APPL% value is much smaller during this time interval. But that's all what we can see. On the other hand we see

many more CPU delays during the capping period **5.**. usually we would be alarmed to see much higher CPU delays but they are not reflected in the CICS transaction response time. So all we can state at this point is that CPU delay are not necessarily sufficient to identify a workload problem. In our case it is much better to watch the average and percentile CICS response times in order to understand whether the workload performs well or not and what impact a configuration change really has on goal achievement.

A.4. Summary

To summarize this little excursion we should remember:

First:
Execution Velocities and execution velocity goals represent system characteristics. The metric is very dependent on the number of execution units and processors they can use. Therefore the metric is also very sensitive to changes in system configuration.

Second:
A response time goal reflects much better the behavior of transactional workloads. This is true in both directions. If a change to the system does not hamper the workload it does not reflect this change. If something else happens it is reflected in the response time and points us to events which we should really analyze.

Third:
Whenever possible use a response time goal to manage on-line transactional workloads. There are exceptions to this rule as we saw in chapter 4 but in general you should use response time goals.

B. A Bad Example for Defining CICS Service Classes

In this addendum we will give an example of what may happen if too many service classes are defined for transactions which share the same set of regions.

B.1. Service Class Definitions

Workload	Service Class	Per	Imp	Printed
CICS	MOSTTRAN	1	2	80% complete within 00:00:00.080
CICS	SPECTRAN	1	2	80% complete within 00:01:40.000
CICS	SYSTRAN	1	3	80% complete within 00:00:01.000
STC	REGIONS	1	3	Execution velocity of 50
CICS	DEFTRAN	1	4	50% complete within 00:00:02.000
CICS	LONGTRAN	1	4	50% complete within 24:00:00.000

Figure B.1.: Service Classes for a CICS workload

The table in figure B.1 shows an excerpt of a service definition with 5 external service classes for CICS transactions and one service class for the CICS regions. The name of the service classes already indicate the type of the transactions:

- MOSTTRAN is the service class for the majority of the CICS transactions. It also has the most stringent goal definition
- SPECTRAN is a service class which assumes longer running but also important transactions
- SYSTRAN is for internal CICS transactions
- DEFTRAN is the default service class for CICS work. All transactions which do not meet a classification rule land in this class
- LONGTRAN is a service class for very long running CICS work

B.2. Internal Service Classes

In chapter 2 we showed that up to 31 internal service classes could be created for 5 external service classes. Fortunately this is very seldom the case. But we will see that much fewer internal service classes show already surprising results.

For analyzing the internal service classes we take a look at the WLM SMF 99 data for the CICS transactions. For each internal service classes WLM tracks the number of observations. This is the number of times when WLM recognized a Performance Block (PB) being associated with a transaction of an

external service class. WLM samples the PBs every 250ms, so it is possible that for long running transactions multiple observations occur for the same transaction and that for very short running transactions not all transactions will be captured. It is also possible to observe an internal service class with zero observations. This still means that very few transactions run in this service class as opposed to entries without a number .Figure B.2 lists the internal service classes as rows and the external service classes as columns. The numbers are the observations. The orange marked entries are the internal service classes for which most observations have been accounted for. In our example we see that 8 internal service classes were created for the five external service classes.

10:38	SYSTRAN	DEFTRAN	MOSTTRAN	SPECTRAN	LONGTRAN	
$SRMS028	17		19091		83	
$SRMS029	67		851			
$SRMS02A	0		0	0	0	
$SRMS02B	0	0	0			
$SRMS02C						
$SRMS02D	24	1	19857		66	
$SRMS02E	28	1	20061	5102	61	
$SRMS030	0	0	0	0		
Total Observations	136	2	59860	5102	210	65100

Figure B.2.: Internal to external service class mapping at 10:38

We can easily observe that the majority of all work is associated with the external service class MOSTTRAN (around 92%). Only for SPECTRAN we can also observe a significant number of observations (around 7.8%). For the other three external service classes only very few observations are accounted. Nevertheless these definitions have an effect. The mixture of transactions cause that WLM creates 3 internal service classes for regions ($SRMS028, $SRMS02D, and $SRMS02E) which process nearly the same amount of transactions for the external service class MOSTTRAN.

Observation 1:
Service classes with low number of transactions (or observations) can result in splitting the regions which process basically all the same type of transactions into different internal service classes.

B.3. Changing Internal Service Classes

As a second step we will take a look at different time intervals of the same topology. We can see that the topology in figure B.3 has changed twice:

- At 10:42 the internal service class $SRSMS028 shows only very little observations so the majority of transactions is now distributed between $SRMS02D and $SRSMS02E
- At 10:47 the situation has completely changed:
- The internal service class$SRMS030 is no longer present but instead the internal service class $SRMS02C has been created. Please note that these 2 service classes do not have the same mixture of transactions
  - MOSTTRAN transactions are now executed in $SRMS028, $SRMS02A, and $SRMS02D
  - SPECTRAN transactions are now mostly executed in $SRMS02A too
  - It seems that the regions which were previously associated with internal service class $SRMS02E no longer process transactions for service class DEFTRAN. Therefore the regions are now associated with the internal service class $SRMS02A

As a result we can observe that the transaction mix and the association of regions with internal service classes may change frequently.

10:42	SYSTRAN	DEFTRAN	MOSTTRAN	SPECTRAN	LONGTRAN
$SRMS028	52		797		1
$SRMS029	35		458		
$SRMS02A	0		0	0	0
$SRMS02B	0	0	0		
$SRMS02C					
$SRMS02D	45	2	39971		160
$SRMS02E	28		20498	5333	69
$SRMS030	0	0	0	0	
Totals	160	2	61725	5333	230

10:47	SYSTRAN	DEFTRAN	MOSTTRAN	SPECTRAN	LONGTRAN
$SRMS028	146		20982		78
$SRMS029	0		0		
$SRMS02A	27		19293	5309	78
$SRMS02B	89	1	1262		
$SRMS02C	0		0	0	
$SRMS02D	13	3	19224		79
$SRMS02E	0	0	0	0	0
$SRMS030					
Totals	276	4	60761	5309	234

Figure B.3.: Internal to external service class mapping at 10:42 and 10:47

Observation 2:
External service classes with low number of transaction executions may cause that the association of regions to internal service classes changes frequently.

B.4. Service Consumption and Dispatch Priorities

By looking at the service consumption for the internal service classes we can observe the changes of region association between different internal service classes.Figure B.4 shows the service consumption by internal service class in the time frame from 10:35 to 10:50. We see that corresponding to the tables if figure B.2 the service consumption switches from $SRMS02E to $SRMS02A and from $SRMS028 to $SRMS02D and backwards. The interesting question is whether this has a noticeable effect?

Figure B.5 answers this question.The following graphic the Dispatch Priorities assigned to the internal service classes and therefore to the CICS regions. We see that internal service classes $SRMS028, $SRMS02A, $SRMS02D and $SRMS02E always show different dispatch priorities even if they all process primarily transactions for MOSTTRAN. We can also observe that internal service classes with very little CPU consumption very often show a much higher dispatch priority than the internal service classes with high CPU consumption. Finally the long running and default transactions share the same regions like the short running transactions and they are treated like them.

Observation 3:
Different internal service classes which process mostly the same type of transactions will receive different access to resources and therefore the transactions will have different access to resources depending on the region they land in.

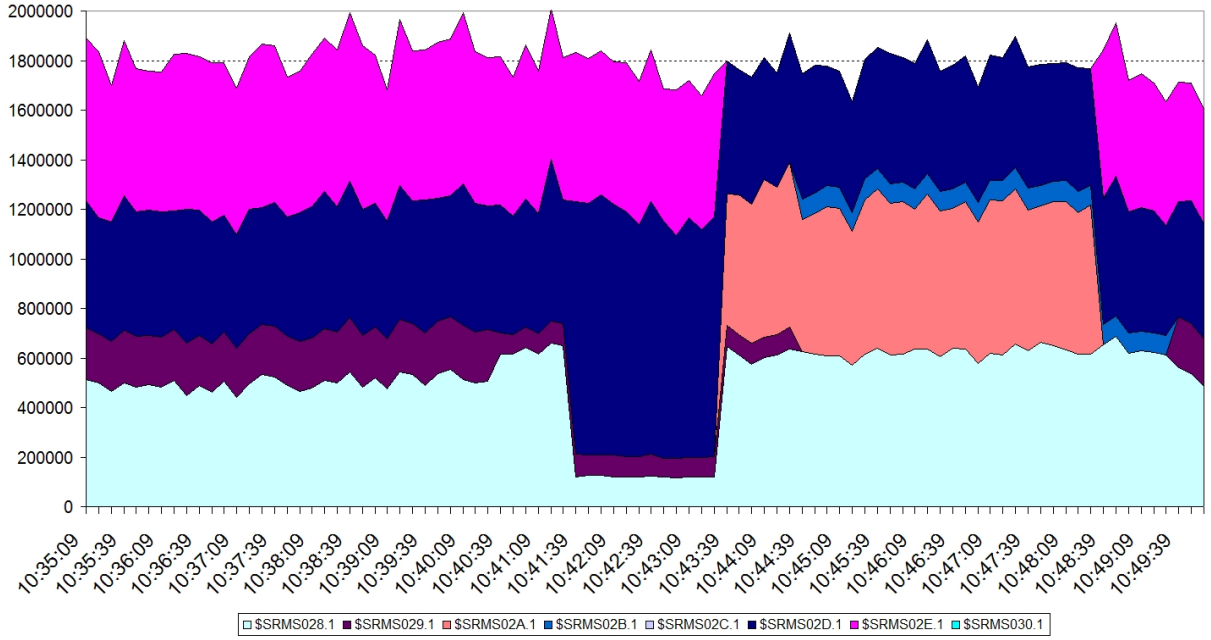


Figure B.4.: Service consumption for internal service classes

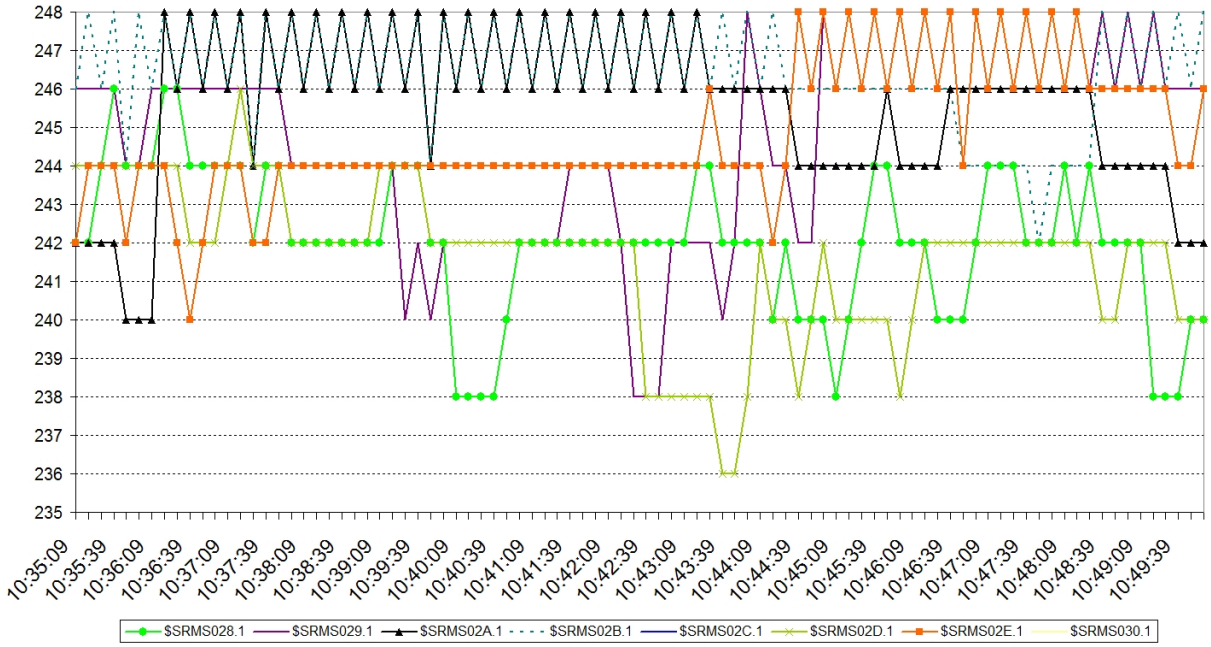


Figure B.5.: Dispatch priorities for internal service classes

Observation 4:
Internal service classes with very small CPU consumption will be treated as low consumers and therefore receive better access to the CPU.

Observation 5:
Long running transactions with very few completions will receive the same access to resources then the majority of the transactions being processed by the regions. Therefore they get a free ride and typically will not be treated worse then the short running high important transactions.

IBM Copyright © 2014

C. Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX®, AS/400®, BatchPipes®, C++/MVS, CICS®, CICS/MVS®, CICSPlex®, COBOL/370, DB2®, DB2 Connect, DB2 Universal Database, DFSMS/MVS®, DFSMSdfp, DFSMSdss, DFSMShsm, DFSORT, e (logo)®, ECKD, ES/3090, ES/9000®, ES/9370, ESCON®, FICON, GDPS, Geographically Dispersed Parallel Sysplex, HACMP/6000, Hiperbatch, Hiperspace, IBM®, IBM (logo)®, IMS, IMS/ESA®, Language Environment®, Lotus®, OpenEdition®, OS/390®, Parallel Sysplex®, PR/SM, pSeries, RACF®, Redbooks, RISC System/6000®, RMF, RS/6000®, S/370, S/390®, S/390 Parallel Enterprise Server, System/360, System/370, System/390®, System z, ThinkPad®, UNIX System Services, VM/ESA®, VSE/ESA, VTAM®, WebSphere®, xSeries, z/Architecture, z/OS, z/VM, zSeries

The following are trademarks or registered trademarks of other companies:

- Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
- Red Hat, the Red Hat "Shadow Man" logo, and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc., in the United States and other countries.
- SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

D. Glossary

A

APPC *Advanced Program to Program Communication* - Protocol for program to program communication between multiple systems

APPL% *Percent of Application Utilization* -Converted application time into an overall percent of a single CPU used for the workload

ASCB *Address Space Control Block* - z/OS control block which represents a virtual storage entity tight to an end user or set of programs to execute

B

BCP *Basic Control Program* - z/OS or MVS kernel routines

C

CCW *Channel Command Word* - Defines an I/O operation (read, write, control) to be performed on an I/O device

CDS *Couple Dataset* - Dataset which contains control information to setup a parallel sysplex environment

CEC *Central Electronic Complex* - The system (processors, memory, I/O adapters), not including I/O devices

CFCC *Coupling Facility Control Code* - Operating System of the coupling facility

CHPID *Channel Path Identifier* - Identification of the channel path, typically a number

CICS *Customer Information Control Server* - A transaction monitor that runs primarily on z/OS

CISC *Complex Instruction Set Computing* - Processing architecture which contains many complex instructions which perform functions like small programs

CKD *Count Key Data* - System z disk architecture

CP *Central Processor* - Standard processor of a System z

CPU *Central Processing Unit* - see CP

CSS *Channel Subsystem* - The heart of moving data in and out of of a mainframe

CSSID *Channel Subsystem Identifier* - Number which identifies the Channel Subsystem in case multiple exist

D

DASD *Direct Access Storage Device* - A storage device which supports direct access (typically a disk)

DB2 *Database* - Relational database based on E. F. Codd's theory of relational databases

DDF *Distributed Data Facility* - Component of DB2 to exchange information with external clients

DEDB *Data Entry Database* - Fast path database for IMS

DL/I *Data Language Interface* - Database of IMS

DRDA *Distributed Relational Database Architecture* - Distributed database architecture of the open group standard

E

ECKD *Extended Count Key Data* - incorporates fixed-block and CKD architecture

EMIF *ESCON Multiple Image Facility* - Feature which allows to use ESCON channels from multiple partitions

ESA/390 *Enterprise Systems Architecture/390* - 32-bit predecessor of System z architecture

ESCON *Enterprise System Connection* - Half-duplex optical fiber serial channel

ESPIE *Extended Specify Program Interruption Exit* - Interrupt exit routine

ESTAE *Extended Specified Task Abnormal Exit* - Recovery routine for z/OS user or problem state programs

ESTI *Enhanced Self-Timed Interface* -

ETR *External Time Reference* - Device to synchronize all TOD (time-of-day) clocks in a cluster environment (Parallel Sysplex)

EXCP *Execute Channel Program* - z/OS macro to execute an I/O operation

F

FCP *Fibre Channel Protocol* - Transport protocol for transporting SCSI commands on Fibre Channel networks

FICON *Fibre Channel Connection* - Full-duplex fibre optical serial channel

FIFO *First In, First Out* - Queuing mechanism

FLIH *First Level Interrupt Handler* - Interrupt handler that gets immediate control when the interrupt occurs (where the new Program Status Word points to)

FRR *Functional Recovery Routine* - Recovery routine for z/OS system programs

G

GDPS *Global Dispersed Parallel Sysplex* - Parallel Sysplex which is spatially distributed to ensure high availability

GRS *Global Resource Serialization* - z/OS subsystem which supports global lock management

H

HCD *Hardware Configuration Dialog* - z/OS component to define I/O devices to the system

HFS *Hierarchical File System* - UNIX file system on z/OS

HMC *Hardware Management Console* - Console to access and manage hardware components of System z

HWA *Hardware Address* -

I

I/O *Input/Output* - Abbreviation for all parts which send data to and from an electronic complex

ICB *Integrated Cluster Bus* - Bus for connecting system in a parallel sysplex for short distance. The bus relies on few parts and provides very high speed and reliable connectivity

ICF *Integrated Coupling Facility* - Processor on system z which allows to run coupling facility control code

IFL *Integrated Facility for Linux* - Processor on system z which allows to execute z/VM and Linux operating systems

IML *Initial Microcode Load* - Initialization process of System z hardware. At its completion, operating systems can be booted (IPLed).

IBM Copyright © 2014

IMS *Information Management System* - A transaction monitor and database for z/OS (introduced 1968 for the Apollo space program)

IOCDS *Input/Output Configuration Data Set* - Data set which contains hardware I/O definitions (related to IODF; also should be consistent with IODF)

IODF *I/O Definition File* - Data file which contains the I/O definitions for System z (created by HCD)

IPC *Inter Process Communication* - Protocol for system processes to interact which each other

IPL *Initial Program Load* - Process to start the z/OS operating system

IRB *Interrupt Request Block* - z/OS Control Structure to start an I/O routine

IRD *Intelligent Resource Director* - A combination of multiple z technologies to enhance the autonomic capabilities of PR/SM, z/OS and the I/O subsystem

IRLM *IBM Resource Lock Manager* - Lock manager for DB2 and IMS

ISPF *Interactive System Productivity Facility* - End user interface for TSO users

J

JES *Job Entry System* - z/OS subsystems which support the execution of scheduled programs

L

LCP *Logical Processor* - Representation of a processor to the virtual system or logical partition

LCSS *Logical Channel Subsystem* - A system may use multiple logical channel subsystems (currently up to 4) to increase connectivity

LDAP *Lightweight Directory Access Protocol* - Application protocol for accessing and maintaining distributed directory information services over an IP network

LIC *Licensed Internal Code* - System z microcode or firmware

LICCC *Licensed Internal Code Configuration Control* -

Ln *Level n Cache* - L1 is closest to the processor, the highest number is used to describe memory (”storage” in System z terminology)

LPAR *Logical Partition* - Container which hosts an operating system to execute on System z virtualization layer. Up to 60 LPARs are supported

M

MBA *Memory Bus Adapter* - I/O hub chip used on z10 and earlier machines. No longer used on z196.

MCM *Multi-Chip Module* - contains processor and storage controller chips

MCSS *Multiple-logical-Channel Subsystem* - Restructuring of the physical CSS into multiple logical instances in order to enable more external devices to be attached to the CEC

MLCS *Multiple Logical Channel Subsystems* - see MCSS

MMU *Memory Management Unit* - Hardware component which handles virtual memory

MPL *Multi Programming Level* - Term which expresses the ability of workload to access system resources

MSU *Million of Service Units per Hour* - Unit to measure CPU capacity on System z

MTTW *Mean Time To Wait* - Algorithm which gives access to units of work based on their deliberate wait time

MVS *Multiple Virtual Storage* - Original name of z/OS based on the ability to support multiple applications in virtual storage

IBM Copyright © 2014

O

OLTP *Online Transaction Processing* - Umbrella term for transaction processing

OSA *Open System Adapter* - Networking adapter

P

PAV *Parallel Access Volume* - Protocol which supports parallel access to the same I/O device

PCHID *Physical Channel Identifier* - identifies the plug position of a channel adapter

PCP *Physical Processor* - see CP

PPRC *Peer to Peer Remote Copy* - A protocol to replicate a storage volume to a remote site

PR/SM *Process Resource and System Manager* - Management component of the logical partition technology of System z (alias for LPAR hypervisor)

PSW *Program Status Word* - Central register to control all program execution

PU *Processing Unit* - Physical processor

Q

QDIO *Queued Direct I/O* - memory to Memory I/O mechanism between LPARs on System z

R

RACF *Resource Access Control Facility* - z/OS subsystem which supports access control

RAS *Reliability, Availability, Serviceability* - Terminology to depict the robustness of information technology systems (originated from IBM mainframe)

RETAIN *Remote Technical Assistance Information Network* - IBM network to handle service requests for end users

REXX *Restructured Extended Executor* - Interpretive Execution Language from IBM

RISC *Reduced Instruction Set Computing* - Processing architecture which only contains elementary instructions like LOAD, STORE, and register-to-register operations

RLS *Record Level Sharing* - VSAM access method which introduces record sharing and serialization

RMF *Resource Measurement Facility* - z/OS Performance Monitor

RRMS *Resource Recovery Management Services* - z/OS component to synchronize the activities of various syncpoint managers

RSF *Remote Support Facility* - Part of HMC to report and repair hardware and firmware components

S

S/360 *IBM System/360* - Is a mainframe computer system family announced by IBM on April 7, 1964. It is the computer architecture of which System z is the current incarnation.

SAP *System Assist Processor* - System z I/O processor

SCC *Storage Controller Control* - Storage controller chip

SCD *Storage Controller Data* - Cache chip

SCE *Storage Control Element* - Controls access to main storage data by processor unit

SDWA *System Diagnostic Work Area* - Control structure to capture information in case of an abnormal program termination

SE *Support Element* - Laptop that acts as user interface to System z machine

SIE *Start Interpretive Execution* - Instruction to drive a processor in a logical partition (LPAR) or virtual machine (z/VM)

SIGP *Signal Processor* - Instruction to inform a processor about status change

SLE *Session Level Encryption* - Encryption between originator and receiver of data across all network elements

SLIH *Second Level Interrupt Handler* - Term which encompasses a set of specialized interrupt handling routines

SMF *Systems Management Facility* - z/OS component which supports performance and status logging

SMP *Symmetric Multiprocessing* - A computer system with all physical processors accessing the same storage and I/O subsystems

SRB *Service Request Block* - Control structure to execute a z/OS system program

SRM *System Resource Manager* - Component of z/OS for resource management (introduced 1974, now part of WLM)

STP *Server Time Protocol* - Follow-on to ETR

SU/sec *Service Unit per second* - Capability of a System z processor to execute instructions

SVC *Supervisor Call* - Interface to invoke a z/OS system program

Sysplex *System Complex* - A single logical system running on one or more physical systems

System z *IBM mainframe computer brand* - Current 64-bit incarnation of the S/360 architecture

T

TCB *Task Control Block* - Control Structure to execute user or problem state programs on z/OS

TSO *Time Sharing Option* - z/OS component which supports the parallel execution of multiple end users on MVS

U

UCB *Unit Control Block* - z/OS control structure which represents an I/O device

UoW *Unit of Work* - An execution unit on z/OS

USS *Unix System Services* - z/OS component which supports a full functioning UNIX environment on z/OS

V

VCPU *Virtual CPU* - see LCP

VMM *Virtual Machine Monitor* - Hypervisor or control program to run multiple virtual machines

VSAM *Virtual Storage Access Method* - A set of access methods for System z I/O devices

VTAM *Virtual Terminal Access Method* - Access method for communications devices (now part of z/OS TCPIP subsystem)

VTOC *Volume Table of Content* - Index of a DASD device

W

WLM *Workload Manager* - Central z/OS component for resource management (introduced 1995)

IBM Copyright © 2014

X

XCF *Cross System Coupling Services* - z/OS Services which support the exploitation of a z/OS sysplex

XES *Cross System Extended Services* - z/OS services which support the access to the coupling facility

XRC *Extended Remote Copy* - System z protocol for data replication

Z

z114 *zEnterprise 114* - Mid-range end model of System z processor family (2011)

z196 *zEnterprise 196* - High end model of System z processor family (2010)

zEC12 *zEnterprise EC12* - High end model of System z processor family (2012)

zAAP *System z Application Assist Processor* - System z processor to execute Java code. This processor type can only be used by z/OS and only for instrumented software like the Java Virtual Machine. A special instruction tells the dispatcher when Java execute starts and ends.

zFS *System z File System* - UNIX file system on z/OS

zIIP *System z Integrated Information Processor* - System z processor to execute code which is subject to get offloaded from regular processors. The offload capability is described by the middleware through an interface to WLM and the z/OS dispatcher. Exploiters are middleware like DB2 and TCPIP.D5

IBM Copyright © 2014

Bibliography

[1] *TSO Time Sharing Option im Betriebssystem z/OS*, Dr. Michael Teuffel, Oldenbourg, 2002, ISBN-13: 978-3486255607

[2] *Das Betriebssystem z/OS und die zSeries: Die Darstellung eines modernen Grorechnersystems*, Dr. Michael Teuffel, Robert Vaupel, Oldenbourg, 2004, ISBN-13: 978-3486275285

[3] *High Availability and Scalability of Mainframe Environments*, Robert Vaupel, KIT Scientific Publishing, 2013, ISBN-13: 978-3-7315-0022-3

[4] *In Search Of Clusters, The Ongoing Battle in Lowly Parallel Computing*, Gregory F. Pfister, Prentice Hall, 1998, ISBN 0-13-899709-8

[5] *Adaptive Algorithms for Managing A Distributed Data Processing Workload*, J. Aman, C.K. Eilert, D. Emmes, P. Yocom, D. Dillenberger, IBM Systems Journal, Vol. 36, No. 2, 1997, Seiten 242-283

[6] *MVS Performance Management (ESA/390 Edition)*, Steve Samson, J. Ranade IBM Series, Printed and bound by R.R.Donnelley and Sons Company, ISBN 0-07-054529-4, 1992

[7] *z/OS Workload Manager - How it works and how to use it* Robert Vaupel, March 2014, 3rd edition http://www-03.ibm.com/systems/z/os/zos/features/wlm/WLM_Further_Info.html

[8] *Resource Groups and how they work* Dieter Wellerdiek, 2008, http://www-03.ibm.com/systems/z/os/zos/features/wlm/WLM_Further_Info.html

[9] *ABC of z/OS System Programming*, IBM Redbooks, Volume 11, SG24-6327-xx

[10] *OS/390 MVS Parallel Sysplex Capacity Planning*, IBM Redbook, SG24-4680-01, January 1998

[11] *z/OS MVS Planning: Workload Management*, z/OS Literatur, SA22-7602-xx

[12] *System's Programmer Guide to: Workload Management*, IBM Redbook, SG24-6472-xx

[13] *z/OS MVS Programming: Workload Management Services*, z/OS Literatur, SA22-7619-xx

[14] *z/OS Resource Measurement Facility: Performance Management Guide*, z/OS Literatur, SC28-1951-xx

[15] *z/OS Basic Skills Center*, http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp

[16] *CICS Transaction Server for z/OS: Performance Guide*, IBM Literatur, Various numbers please check your CICS version and release

[17] *CICS Transaction Server for z/OS: Customization Guide*, IBM Literatur, Various numbers please check your CICS version and release

[18] *IMS System Administration Guide*, IBM Literatur, Various numbers please check your IMS version and release

Index

IBM. Copyright © 2014