

ABCs of z/OS System Programming

Volume 12

Workload Manager, WLM policy

WLM goal management, WLM
functions

WLM ISPF application



Paul Rogers
Alvaro Salla

Redbooks



International Technical Support Organization

ABCs of z/OS System Programming Volume 12

January 2010

Archived

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

Archived

First Edition (January 2010)

This edition applies to Version 1 Release 11 of z/OS (5694-A01) and to subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2010. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The team who wrote this book	x
Become a published author	x
Comments welcome	x
Chapter 1. Workload Manager	1
1.1 Workload management	2
1.2 WLM traffic analogy	3
1.3 Workload management environment	5
1.4 Service definition	7
1.5 WLM service definition constructs	9
1.6 Workloads, service policies, and service classes	11
1.7 Workloads	13
1.8 Service classes	14
1.9 Service class goals	16
1.10 Service classes in workloads	19
1.11 WLM types of goals	21
1.12 Service units	23
1.13 Execution velocity - discretionary goals	25
1.14 Discretionary goals	27
1.15 Assigning service classes	29
1.16 Work qualifiers	31
1.17 Service classes	32
1.18 Service class period	34
1.19 Subsystems supported by WLM	36
Chapter 2. WLM ISPF application	39
2.1 ISPF administrative application	40
2.2 Defining a service definition	41
2.3 Service policy override	43
2.4 Creating a service definition	45
2.5 Creating a service policy (1)	46
2.6 Creating a service policy (2)	47
2.7 Creating a service policy (3)	49
2.8 New service policy	50
2.9 Creating a workload	51
2.10 Defining service class goals	52
2.11 Creating a service class (1)	54
2.12 Creating a service class (2)	55
2.13 System-provided service classes	57
2.14 Address space dispatch priority	59
2.15 Subsystem list for classification rules	61
2.16 Classification qualifiers to classify work	62
2.17 Classification group qualifiers	64
2.18 External properties for WebSphere transactions	66
2.19 Create classification rules for WebSphere	67

2.20 Classification rules for CICS	68
Chapter 3. WLM goal management	71
3.1 WLM additional functions	72
3.2 WLM functions (1)	74
3.3 WLM functions (2)	76
3.4 WLM performance behavior	78
3.5 Sampling transaction WLM states	81
3.6 Performance Index	83
3.7 Sysplex PI and local PI	85
3.8 PI calculation - average response time goal	87
3.9 PI calculation - execution velocity goal	88
3.10 Percentile response time - PI calculation	89
3.11 PI calculation - discretionary goals	91
3.12 WLM policy adjustment routine	93
3.13 WLM system-provided special service classes	96
3.14 Dispatching priorities assignment	98
3.15 CPU management	100
3.16 Reasons for CPU delays	102
3.17 CPU Critical option	103
3.18 CPU Critical and resource groups	105
3.19 Storage Critical option	106
3.20 Storage Critical in action	107
3.21 I/O management	108
3.22 I/O priority management	110
3.23 Parallel Access Volume	113
3.24 WLM dynamic alias PAV management	114
3.25 HyperPAV	116
3.26 Intelligent Resource Director	118
3.27 Moving resources through IRD	119
3.28 LPAR cluster	121
3.29 RMF LPAR cluster report	122
3.30 Managing CICS and IMS/DC WLM goals	123
3.31 CICS/IMS address space server topology	125
3.32 CICS transaction type of goal	127
Chapter 4. WLM miscellaneous functions	129
4.1 Scheduling environment	130
4.2 Scheduling environment implementation	131
4.3 Dynamic workload balancing	133
4.4 WLM load balancing algorithms	135
4.5 Dynamic workload routing support	136
4.6 WLM dynamic load balancing exploiters	138
4.7 WLM server address space management	140
4.8 Application environment	142
4.9 Dynamic application environment	144
4.10 WebSphere and WLM	145
4.11 WLM-managed initiators (1)	146
4.12 WLM-managed initiators (2)	148
4.13 z/OS UNIX performance overview	150
4.14 WLM in goal mode	151
4.15 Capping and WLM	152
4.16 Resource group capping	154

4.17 WLM resource groups.....	156
4.18 Resource group - type 2	158
4.19 Resource group - type 3	159
4.20 Soft capping	161
4.21 Group capacity capping	163
4.22 WLM discretionary work capping	165
4.23 WLM z10 EC capacity provisioning.....	167
4.24 Capacity provisioning domain.....	169
4.25 Capacity provisioning policy	171
4.26 WLM buffer pool management	173
4.27 WLM buffer pool size management	175
4.28 WLM interfaces	177
4.29 WLM console commands (1)	179
4.30 WLM console commands (2)	181
4.31 Other WLM-related console commands	182
Related publications	183
IBM Redbooks	183
Other publications	183
How to get Redbooks.....	183
Help from IBM	183

Archived

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICSplex®	IMS™	System z10™
CICS®	IMS/ESA®	System z9®
DB2 Connect™	Language Environment®	System z®
DB2®	Lotus®	Tivoli®
Domino®	MQSeries®	VTAM®
DS8000®	NetView®	WebSphere®
ECKD™	Parallel Sysplex®	z/Architecture®
Enterprise Storage Server®	PR/SM™	z/OS®
ESCON®	RACF®	z/VM®
FICON®	Redbooks®	z9®
Geographically Dispersed Parallel Sysplex™	Redbooks (logo)  ®	zSeries®
IBM®	S/390®	
	SOMobjects®	

The following terms are trademarks of other companies:

Novell, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

SAP R/3, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Installations today process different types of work with different response times. Every installation wants to make the best use of its resources, maintain the highest possible throughput, and achieve the best possible system responsiveness. You can realize such results by using workload management. This IBM® Redbooks® publication introduces you to the concepts of workload management utilizing Workload Manager (WLM).

Workload Manager allows you to define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work. The system matches resources to the work to meet those goals, and constantly monitors and adapts processing to meet the goals. This reporting reflects how well the system is doing compared to its goals, because installations need to know whether performance goals are being achieved as well as what they are accomplishing in the form of performance goals.

The ABCs of z/OS® System Programming is a thirteen-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

The volumes contain the following content:

- ▶ Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, Language Environment®, and SMP/E
- ▶ Volume 3: Introduction to DFSMS, data set basics, storage management hardware and software, VSAM, system-managed storage, catalogs, and DFSSMStvs
- ▶ Volume 4: Communication Server, TCP/IP and VTAM®
- ▶ Volume 5: Base and Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), Global Resource Serialization (GRS), z/OS system operations, Automatic Restart Management (ARM), and Geographically Dispersed Parallel Sysplex™ (GPDS)
- ▶ Volume 6: Introduction to security, RACF®, digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries® firewall technologies, LDAP, Enterprise Identity Mapping (EIM), and firewall technologies
- ▶ Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central
- ▶ Volume 8: An introduction to z/OS problem diagnosis
- ▶ Volume 9: z/OS UNIX® System Services
- ▶ Volume 10: Introduction to z/Architecture®, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC
- ▶ Volume 11: Capacity planning, performance management, RMF, and SMF
- ▶ Volume 12: WLM
- ▶ Volume 13: JES3

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS. Before joining the ITSO 21 years ago, he worked in the IBM Installation Support Center (ISC) in Greenford, England providing and JES support for IBM EMEA and the Washington Systems Center. Paul has worked for IBM for 41 years.

Alvaro Salla is an IBM retiree who worked for IBM for more than 30 years, specializing in large systems. He has co-authored many IBM Redbooks publications and spent many years teaching about large systems from S/360 to S/390®. He has a degree in Chemical Engineering from the University of Sao Paulo, Brazil.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:
ibm.com/redbooks
- ▶ Send your comments in an e-mail to:
redbooks@us.ibm.com
- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Workload Manager

One of the strengths of the mainframe platform (zSeries, z9®, and z10, together with the z/OS operating system) in a commercial environment is the ability to run multiple workloads at the same time (with thousands of concurrent transactions) within one z/OS image or across multiple images in a Parallel Sysplex. These workloads may be of differing importance to the business and have differing needs for system resources. As a consequence, they will have different priorities when queuing for such system resources.

The z/OS component that makes this possible is the Workload Manager (WLM). Among several functions, WLM implements dynamic workload balancing, which is an intelligent distribution of the arriving transactions across the z/OS systems.

The idea behind z/OS Workload Manager is to make a “contract” between the installation (your company) and the operating system. The installation classifies the transactions running on the z/OS operating system in distinct service classes. Each service class defines goals for the transaction that express the expectation of how the transaction should perform. WLM uses these goal definitions to manage the transaction across all systems of a sysplex environment. Figure 1-1 on page 2 illustrates the concepts of workload management.

1.1 Workload management

- ❑ MVS workload management provides:
 - A solution for managing workload distribution, workload balancing, and distributing resources to competing workloads
- ❑ MVS workload management is the combined cooperation of various MVS subsystems
 - CICS, IMS/ESA, JES, APPC
 - TSO/E, z/OS UNIX System Services
 - DDF, DB2, SOM, LSFM
 - Internet Connection Server
- ❑ Performance goals

Figure 1-1 Workload management

Workload management

Before the introduction of MVS workload management, MVS required you to translate your data processing goals from high-level objectives about what work needs to be done into the extremely technical terms that the system can understand. This translation not only requires high skill-level staff but also can be protracted and error-prone, and eventually in conflict with the original business goals. Multi-system, sysplex, parallel processing, and data sharing environments add to the complexity. MVS workload management provides a solution for managing workload distribution, workload balancing, and distributing resources to competing workloads. MVS workload management is the combined cooperation of various subsystems (CICS®, IMS/ESA®, JES, APPC, TSO/E, z/OS UNIX System Services, DDF, DB2®, SOM, LSFM, and Internet Connection Server) with the MVS workload management (WLM) component.

Performance goals

Installations today process different types of work with different completion and resource requirements. Every installation wants to make the best use of its resources and maintain the highest possible throughput and achieve the best possible system responsiveness. Workload management makes this possible. With workload management, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to it to meet the goal. WLM will constantly monitor the system and adapt processing to meet the goals.

1.2 WLM traffic analogy

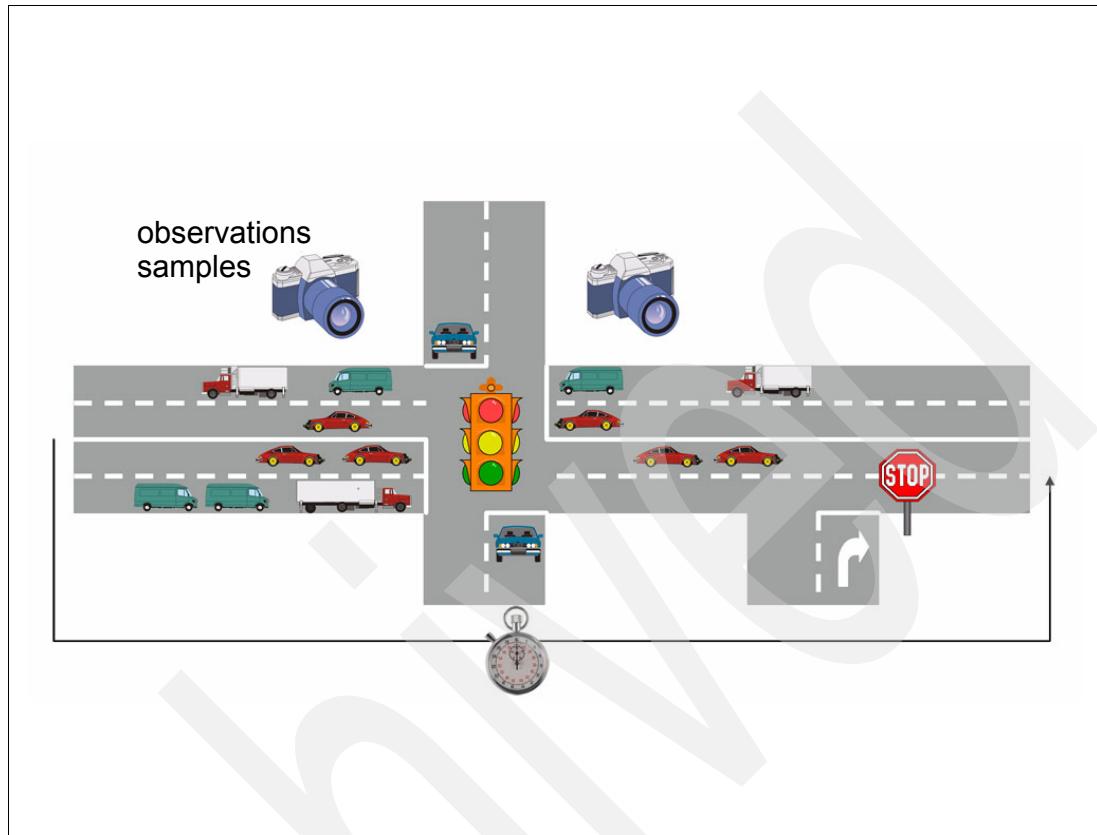


Figure 1-2 WLM traffic analogy

WLM traffic analogy

Figure 1-2 uses a traffic analogy to illustrate how WLM enforces sysplex goals for transaction types.

When you drive a car between two locations, the time it takes is determined by the average speed that you are allowed to drive, the amount of traffic on the streets, and the traffic regulations at crossings and intersections. Based on this, you can identify the time as the basic measure to go from a start to a destination point; you can also see that the amount of traffic is the determining factor in how fast you can travel between these two points. Although the driving speed is usually regulated and constant, the number of cars on the street determines how long you have to wait at intersections and crossings.

As a result, you can identify the crossings, traffic lights, and intersections as the points where you encounter delays on your way through the city. A traffic management system can use these considerations to manage the running traffic. For example, dedicated lanes can allow buses and taxis to pass the waiting traffic during rush hours and therefore travel faster than passenger vehicles. This is a common model of prioritizing a certain type of vehicle against others in a traffic system.

Using this example, you see that it is possible to use time as a measure between two points for driving in a city. You also identified the points where contention may occur, and found a simple but efficient method of prioritizing and managing the traffic.

These concepts can be easily ported to a system; all you need is a technique to accomplish the following tasks:

- ▶ Identify the transaction running in the system
- ▶ Measure the time it runs
- ▶ Determine the contention points

Transaction managers

In a system, the identification of a transaction is supported by middle ware (transaction managers) and the z/OS; they tell WLM when a new transaction enters the system and when it leaves the system. WLM provides constructs to separate the transaction into distinct classes (service classes). These constructs are known as *classification rules*. Classification rules allow an installation to deal with different types of transaction running on the system.

Contention occurs when several transactions want to use the same system resource, including CPUs, I/O devices and storage, and also software constructs like processes (dispatchable units) or address spaces. These resources provide the capability to execute programs and serialization points that allow the programs to access resources or serialized control areas keeping data integrity. WLM monitors some resources (not all of them) to be able to understand how many resources each transaction would require or will wait for.

Service classes

The classification rules and the WLM observation of how the transaction uses the resources provide the base for managing the system through transaction priorities. This management is performed based on the goals the installation defines for the transaction in the service class. After classifying the transactions into distinct service classes, the installation associates a goal with each service class; the goal determines how much service the transaction in the service class is able to receive resource by setting different values of priorities.

Business importance

In addition to the goal, it is also necessary to define how important is to achieve one service class transaction goal when compared with other transaction goals. In the case where several service classes do not achieve their target goals, the *importance* will assist in deciding which service class should be helped in getting resources (in other words, have a priority raise).

1.3 Workload management environment

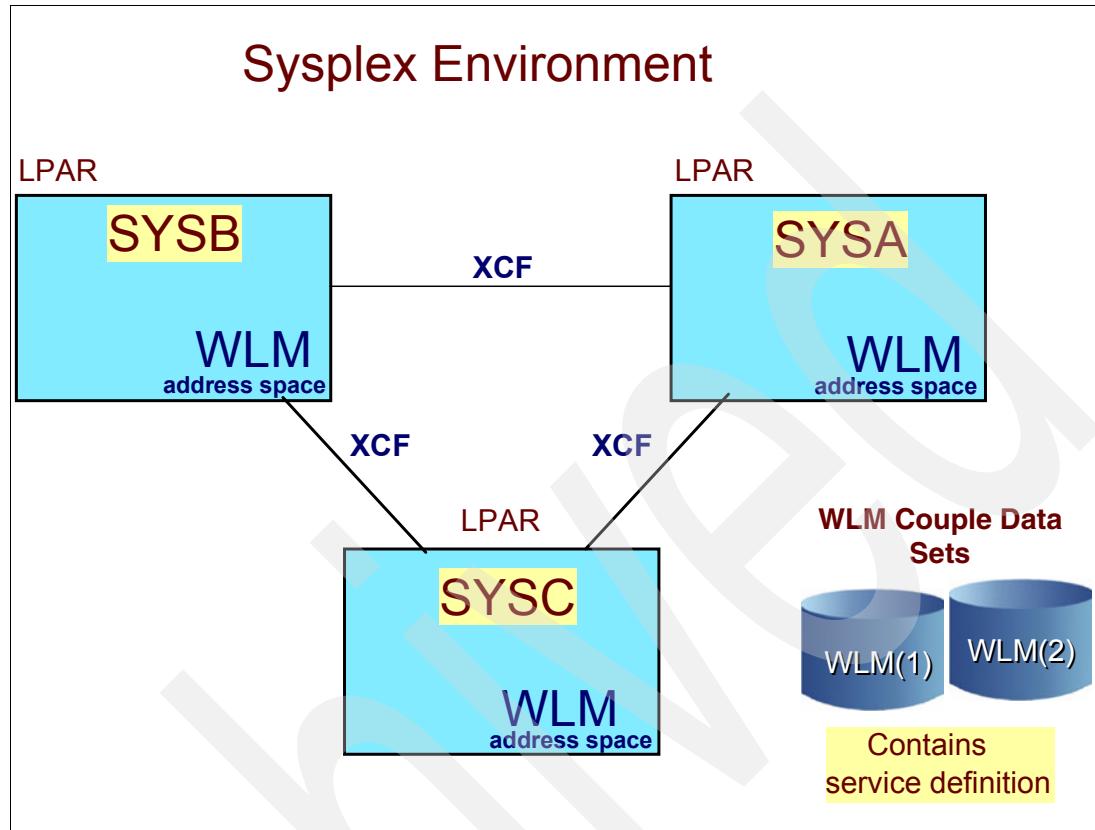


Figure 1-3 WLM address spaces in a sysplex environment

WLM address space

A WLM address space exists in each system in a sysplex, as illustrated by Figure 1-3. The WLM address spaces communicate with each other using XCF services by joining an XCF group (SYSWLM) as each system joins the sysplex.

An XCF group is a set of related members that a multisystem application defines to XCF. A *member* is a specific function, or instance, of the application. A member resides on one system and can communicate with other members of the same group across the sysplex. Communication between group members on different systems occurs over the signaling paths that connect the systems; on the same system, communication between group members occurs through local signaling services.

WLM couple data set

A WLM couple data set is defined for storing the service definition information. If you are running a sysplex with mixed release levels, then format the WLM couple data set from the highest level system. This allows you to use the current level of the WLM application. You can continue to use the downlevel WLM application on downlevel systems if you do not attempt to exploit new function.

Note: The WLM couple data set should be duplexed; an alternate copy is used for backup,

To allocate the WLM couple data set you can either use the facility provided in the WLM application, or you can run an XCF utility. For each case, you need to estimate how many workload management objects you are storing on the WLM couple data set.

Service definition

The service definition, as illustrated in Figure 1-4 on page 7, contains all the information about the installation that is needed for workload management processing. There is one service definition for the entire sysplex. The service level administrator specifies the service definition through the WLM administrative application. The service level administrator sets up “policies” within the service definition to specify the goals for work. A service level administrator must understand how to organize work and be able to assign it performance objectives.

Archived

1.4 Service definition

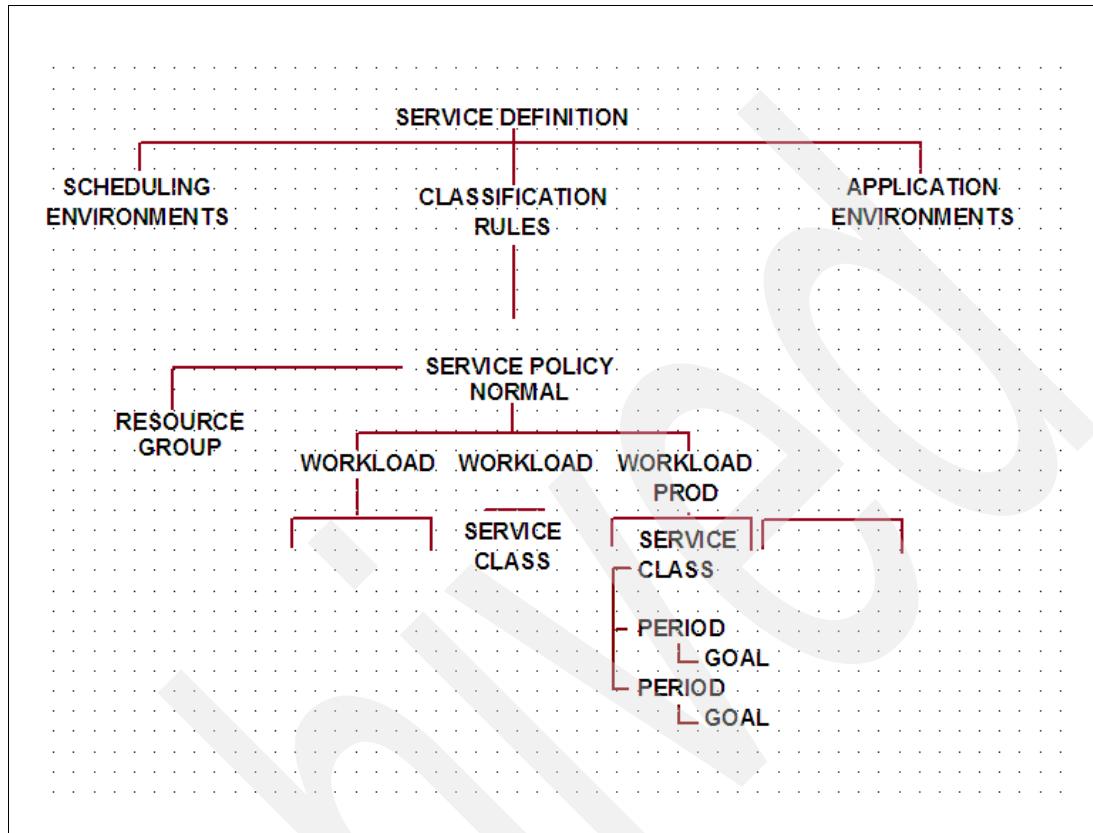


Figure 1-4 Service definitions

Service definition

When you set up your service definition, you identify the workloads, the resource groups, the service classes, the service class periods, and the goals based on your performance objectives. Then you define classification rules and one or more service policies. This information makes up the base service definition.

The service definition contains one or more service policies with constructs:

- ▶ Service classes
- ▶ Workloads
- ▶ Classification rules

Optional service policy optional constructs:

- ▶ Report classes
- ▶ Resource groups
- ▶ Application environments
- ▶ Scheduling environments

ISPF administrative application

A *service definition* consists of one or more service policies.

Service policy

A *service policy* is a named set of overrides to the goals in the service definition. One or more service policies can be defined. When a policy is activated, the overrides are merged with the service definition. You can have different policies to specify goals intended for different times. Service policies are activated by an operator command, or through the ISPF administrative application utility function. A policy applies to all of the work running in a sysplex. Because processing requirements change at different times, service level requirements may change at different times. If you have performance goals that apply to different times or a business need to limit access to processor capacity at certain times, you can define multiple policies.

To start workload management processing, you must define at least one service policy. You can activate only one policy at a time.

Service classes

Service classes are subdivided into periods and group work with similar performance goals, business importance, and resource requirements for management and reporting purposes. You assign performance goals to the periods within a service class.

Workloads

Workloads are a set of service classes together for reporting purposes. All work running in the installation is divided into workloads. Your installation may already have a concept of workload. A *workload* is a group of work that is meaningful for an installation to monitor. For example, all the work created by a development group could be a workload, or all the work started by an application, or in a subsystem.

Classification rules

Classification rules determine how to assign incoming work to a service class and report class. Classification of work depends on having the rules defined for the correct subsystem type. Classification rules are the rules you define to categorize work into service classes, and optionally report classes, based on work qualifiers. A *work qualifier* is what identifies a work request to the system. The first qualifier is the subsystem type that receives the work request.

Report classes

Report classes group work for reporting purposes. They are commonly used to provide more granular reporting for subsets of work within a single service class.

Business importance of work goals

When there is insufficient capacity for all work in the system to meet its goals, business importance is used to determine which work should give up resources and which work should receive more resources. You assign an importance to a service class period, which indicates how important it is that the goal be met relative to other goals. Importance plays a role only when a service class period is not meeting its goal. There are five levels of importance: lowest (5), low (4), medium (3), high (2), and highest (1).

1.5 WLM service definition constructs

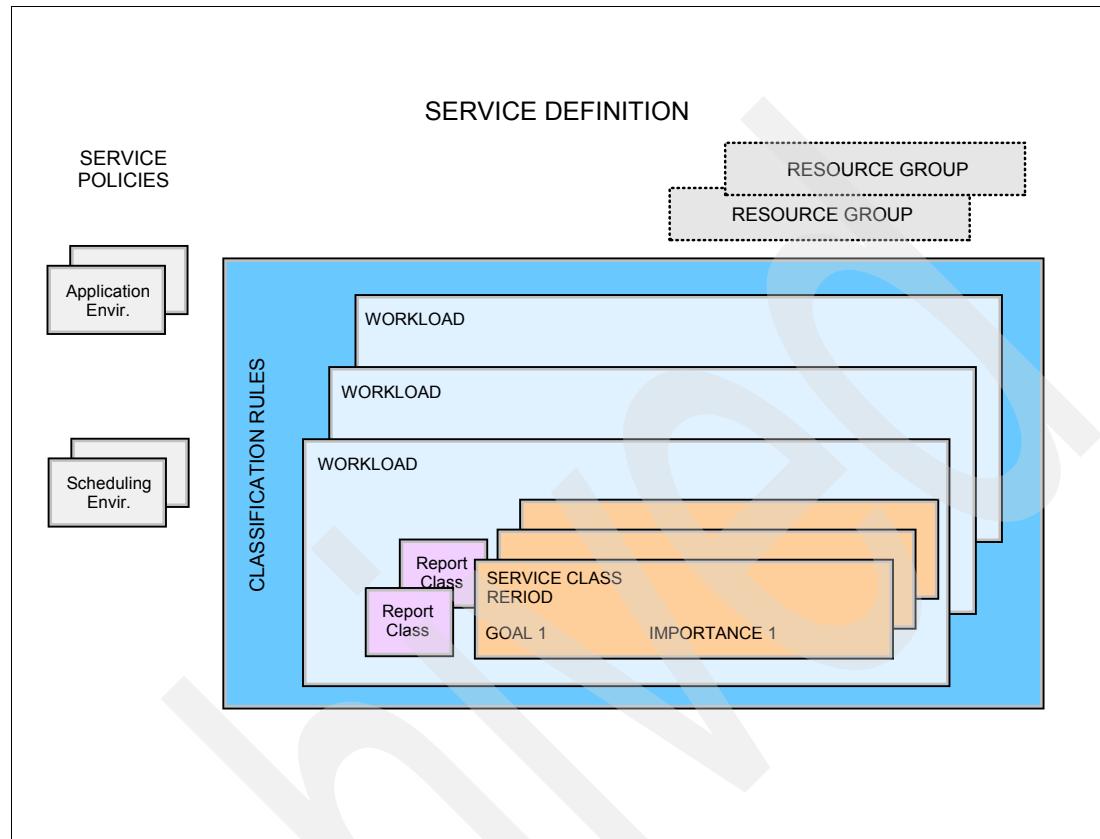


Figure 1-5 WLM constructs

WLM constructs

The service definitions and WLM constructs used for performance management are shown in Figure 1-5. Using an ISPF application, the installation defines a service definition that is stored in a WLM couple data set. The service definition is composed of service policies. If the goals or the importance of your transactions should vary during the day, the month or the year, it is good practice to use several policies. However, you can activate only one policy at a time.

The service policy is comprised of workloads, and these workloads are defined only for accounting purposes. A workload is a named collection of work to be reported as a unit. You can arrange workloads by subsystem (CICS, IMS™) or by major application (Production, Batch, Office). Logically, a workload is a collection of service classes. For each workload, you must have at least one service class that is composed of service class periods that indicate the goal and the importance of the associated transaction. There are also classification rules to link the external properties of a transaction to a service class. Optionally there are other constructs, as such:

- ▶ Resource groups
- ▶ Application environments
- ▶ Scheduling environments

The following sections explain each construct in more detail.

Application environments

Application environments are groups of application functions that execute in server address spaces and can be requested by a client. Workload management manages the work according to the defined goal, and automatically starts and stops server address spaces as needed.

Scheduling environments

Scheduling environments are lists of resource names along with their required states. If an MVS image satisfies all of the requirements in a scheduling environment, then units of work associated with that scheduling environment can be assigned to that MVS image.

Resource groups

Resource groups define processor capacity boundaries across the sysplex. You can assign a minimum and maximum amount of CPU service units per second to work by assigning a service class to a resource group.

1.6 Workloads, service policies, and service classes

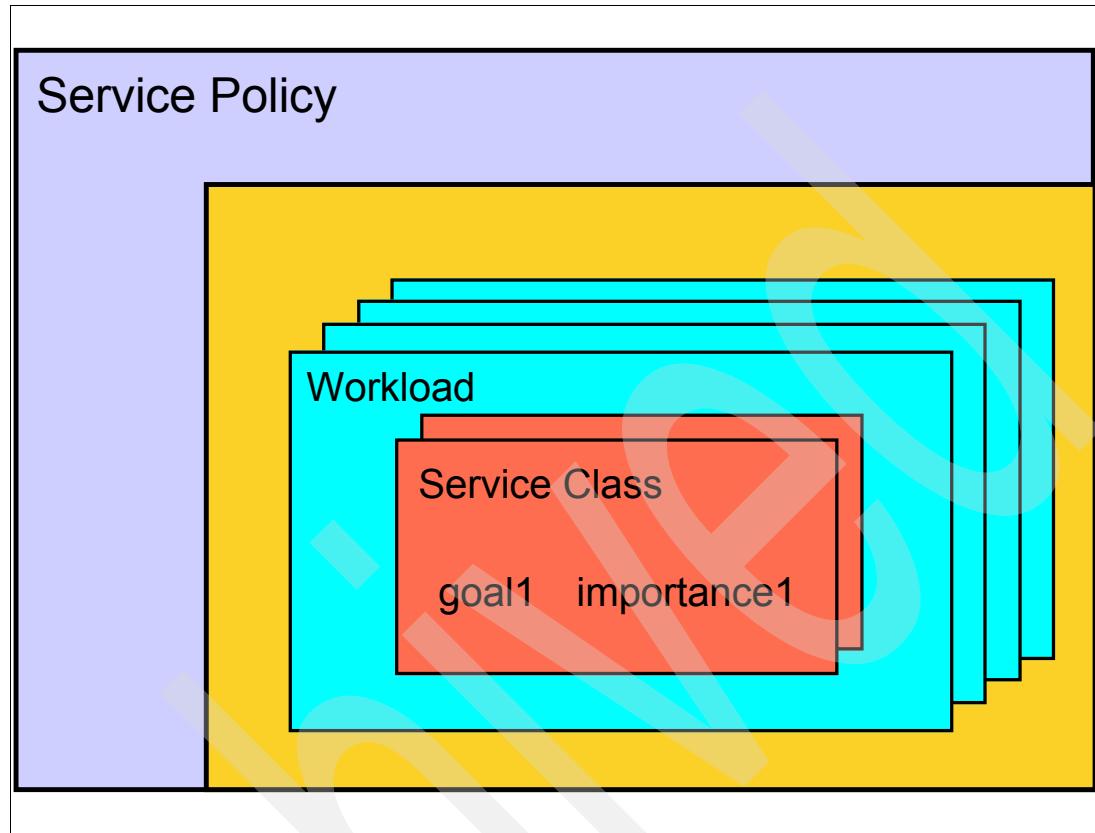


Figure 1-6 Workloads and service classes

Service policy

Figure 1-6 illustrates the relationship between workloads and service classes. A service policy is a named set of overrides to the performance goals and processing capacity boundaries in the service definition. A policy applies to all of the work running in a sysplex. Because processing requirements change at different times, service level requirements may change at different times. If you have performance goals that apply to different times or a business need to limit access to processor capacity at certain times, you can define multiple policies. To start workload management processing, you must define at least one service policy. As previously mentioned, you can activate only one policy at a time.

The active policy can be switched by the following MVS console command to reflect changes in performance objectives; it is a sysplex-wide command:

```
V WLM,POLICY=policyname[,REFRESH]
```

Workloads

A workload is a named collection of service classes to be accounted and reported as a unit. Note that the workload concept does not affect the management of transaction. You can arrange workloads by subsystem (CICS, IMS), by major type of service (production, batch, office), or by line of business (ATM, inventory, department).

The RMF workload Activity report groups performance data by workload and also by service class periods within workloads.

All work running in the installation is divided into workloads. A workload is just a name you give to types of work that run in the sysplex. As named groups of work, workloads are meaningful for an installation to monitor. For example, all the work created by a development group could be a workload, or all the work started by an application, or in a subsystem. Other examples of workloads include batch, DB2, IMS, CICS, OMVS, and so on. The workload name is installation-determined; for example, you can call the IMS work by a workload name of IMS.

Service classes

Service classes are specified into periods. Work is grouped with similar performance goals, business importance, and resource requirements for management and reporting purposes. Goals are assigned to the periods within a service class.

When any kind of work enters the system, such as transactions or batch jobs, this work is classified by WLM by assigning the new work a service class.

Note: WLM allows no more than 100 service classes to be defined in a service definition. The default, however, is 128. This will set aside as much space as you will ever need for service classes, as well as a little extra for other WLM objects.

Service class goals

With workload management, for each service class a performance goal is assigned, and for each performance goal a business importance is assigned that consists of a number from 1 to 5.

You define the goals for work in business terms; that is, importance 1 work is the highest priority work. The system then decides how much resource, such as CPU and storage, should be given to the work to meet its goal.

1.7 Workloads

Workload Selection List					
		Row 1 to 8 of 8			
Command ==> _____					
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, /=Menu Bar					
Action	Name	Description		---Last Change---	
—	BATCH	Batch Workloads		User IBMUSER	Date 2001/08/13
—	DATABASE	Database Workloads		User IBMUSER	Date 2001/08/13
—	EWLM	EWLM Workloads		User FISCHER	Date 2006/12/06
—	ONLINES	Online Workloads		User IBMUSER	Date 2001/08/13
—	STCTASKS	System Workloads		User IBMUSER	Date 2001/08/13
—	TSO	TSO Workloads		User IBMUSER	Date 2001/08/13
—	USS	Unix System Services		User FISCHER	Date 2006/12/07
—	WLZVM	workload for zvm		User ALVARO	Date 2004/08/10
***** Bottom of data *****					

Figure 1-7 Workloads

Workloads

The definition of workloads and performance goals is a part of a service definition. Workloads aggregate a set of service classes together for reporting purposes. To workload management, “work” is a demand for service, such as a batch job, an APPC, CICS, DB2, or IMS transaction, a TSO/E logon, a TSO/E command, or an SOM request. All work running in the installation is divided into workloads.

Figure 1-7 displays a Workload Selection List. In the figure, the workloads defined in the service policy are comprised of a named collection of work to be reported as a unit. As mentioned earlier, you can arrange workloads by subsystem (CICS, IMS), by major application (TSO, batch, office) or by line of business (UNIX System Services, Inventory, department). Logically, a workload is a collection of service classes. Every workload name must be unique for all defined workloads in a service definition.

Note: A workload logically consists of a group of one or more service classes. You associate a workload with a service class in the Service Class panel. Enter your workloads *before* creating your service classes.

1.8 Service classes

<input type="checkbox"/> Service class name and description
➤ Service class period (#)
➤ Service class duration
➤ Service class importance
➤ Service class goal description

Service class XYZ - Batch workload

#	Duration	Imp	Goal description
1			
2			

Figure 1-8 Service classes

Service classes

A service class, as illustrated in Figure 1-8, is a named group of work within a workload with the following similar performance characteristics:

- ▶ Performance goals
- ▶ Resource requirements
- ▶ Business importance to the installation

Service class period

Workload management manages a service class period as a single entity when allocating resources to meet performance goals. A service class can be associated with only one workload. You can define up to 100 service classes. You can assign the following kinds of performance goals to service classes:

- ▶ Average response time goal
- ▶ Response time with percentile goal
- ▶ Velocity goal
- ▶ Discretionary goal

Goal importance

You assign an importance level to the performance goal. Importance indicates how vital it is to the installation that the performance goal be met relative to other goals.

Performance periods

Because some work has variable resource requirements, workload management provides performance periods where you specify a series of varying goals and importances. You can define up to eight performance periods for each service class. You can also assign a service class to a resource group if its CPU service must be either protected or limited.

1.9 Service class goals

- ❑ There are three kinds of goals:
 - Response-time goal types
 - Average response time
 - Percentile response time
 - Execution velocity
 - Discretionary
- ❑ Goal Importance
 - A number from 1 to 5
 - 1 is the most important
- ❑ Performance index

Figure 1-9 WLM performance goals and goal importance

WLM performance goals

Performance goals are assigned to each service class period and you assign an importance to each goal, as displayed in Figure 1-9. *Importance* indicates how important it is to your business that the performance goal be achieved.

Response time goals

Response time goals indicate how quickly you want your work to be processed; that is, the expected amount of time required to complete the work submitted under the service class, in milliseconds, seconds, minutes and hours. Specify either an average response time, or response time with a percentile. That is, workload management does not control all aspects of system performance, so response time scope is confined to the time during which workload management has control of the work. This time includes the time the work is using or waiting for CPU, storage, or I/O service. The response time goals are as follows:

- ▶ Average response time goals are for transactions completing within the period in terms of hours, minutes, and seconds. Response time varies from 15 milliseconds to 24 hours.
- ▶ Response time percentile goals are a goal for work that is to be completed in the specified amount of time. Percentile is the percentage of work in that period that should complete within the response time. Percentile must be a whole number. You must specify a system response time goal, not “end-to-end.” Percentile boundaries vary from 1 to 99.

Note: Workload management does not delay work, or limit it, to achieve the response time goal when extra processing capacity exists.

Execution velocity goals

Because response time goals are not appropriate for all kinds of work, such as long-running batch jobs, there are execution velocity goals. Execution velocity goals define how fast work should run when ready, without being delayed for processor, storage, I/O access, and queue delay. Execution velocity goals are intended for work for which response time goals are not appropriate, such as started tasks or long-running batch work.

This goal is a measure of how fast work should run when ready, without being delayed for WLM-managed resources. Velocity is a percentage from 1 to 99.

Discretionary goals

Discretionary goals are for low priority work for which you do not have any particular performance goal. Workload management then processes the work using resources that are not required to meet the goals of other service classes. Defining a response time goal may not be appropriate for some types of batch work, such as jobs with very long execution times.

Goal importance

Importance is a reflection of how important it is that the service class goal be achieved. Workload management uses importance only when work is not meeting its goal. Importance indicates the order in which work should receive resources when work is not achieving its goal. Importance is required for all goal types except discretionary. Importance applies on a performance period level and you can change importance from period to period. Importance is assigned in five levels, from 1 to 5, with 1 being the highest importance.

Note: When there is insufficient capacity for all work in the system to meet its goals, business importance is used to determine which work should give up resources and which work should receive more. You assign an importance to a service class period, which indicates how important it is that the goal be met relative to other goals. Importance plays a role only when a service class period is not meeting its goal.

Performance index

The performance index (PI) indicates the achievement of the WLM-defined goals for the service class. The performance index is a calculation of how well work is meeting its defined goal.

- ▶ For work with response time goals, the performance index is the actual divided by the goal.
- ▶ For work with velocity goals, the performance index is the goal divided by the actual.

A performance index of 1.0 indicates the service class period is exactly meeting its goal. A performance index greater than 1 indicates the service class period is missing its goal. A performance index less than 1.0 indicates the service class period is beating its goal. Work with a discretionary goal is defined to have 0.81 as a performance index.

Special service classes

A system address space (AS) is assigned to specific and predefined service classes to guarantee adequate system performance:

- ▶ SYSTEM
- ▶ SYSSTCx (x is a number from 1 to 5 or blank)

SYSTEM and SYSSTCx can also be assigned to address spaces or transactions through the classification rules. However, this assignment is not recommended

Archived

1.10 Service classes in workloads

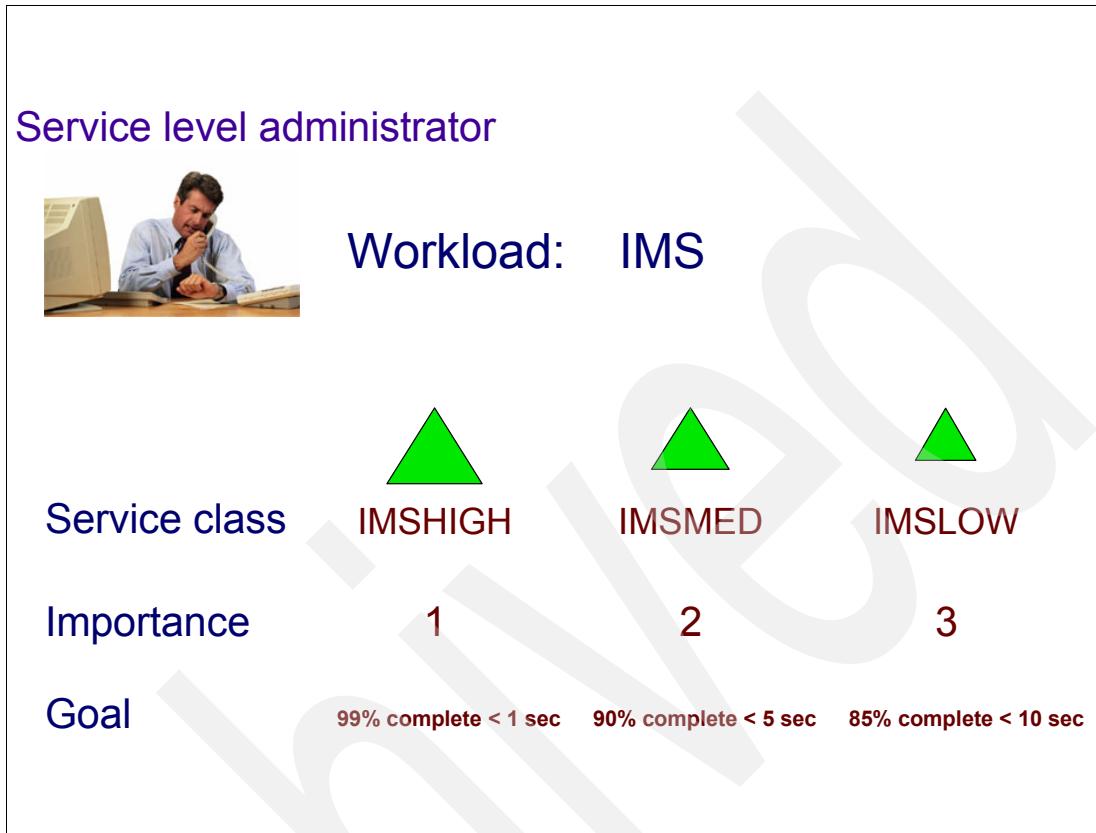


Figure 1-10 Service classes in workloads

Business goals

As previously mentioned, with workload management you define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work to meet its goal.

Note: Figure 1-10 shows a percentile response time goal, which is the percentage of work in that period that should complete within the response time. The percentile must be a whole number.

The figure shows three different service classes defined in the same workload.

An installation should know what it expects to accomplish in the form of performance goals, as well as how important it is to the business that each performance goal be achieved. With workload management, you define performance goals for work, and the system matches resources to the work to meet those goals, constantly monitoring and adapting processing to meet the goals. Reporting reflects how well the system is doing compared to its goals.

Performance administration

Performance administration is the process of defining and adjusting performance goals. Workload management introduces the role of the service level administrator. The service level administrator is responsible for defining the installation's performance goals based on

business needs and current performance. This explicit definition of workloads and performance goals is called a service definition. Defining the service definitions is done by using an online ISPF application that is panel-based.

Some installations might already have this kind of information in a Service Level Agreement (SLA). The service definition applies to all types of work, including CICS, IMS, TSO/E, z/OS UNIX System Services, JES, APPC/MVS, LSF, DDF, DB2, SOM, Internet Connection Server (also referred to as IWEB) and others. You can specify goals for all MVS-managed work, whether online transactions or batch jobs. The goals defined in the service definition apply to *all* work in the sysplex.

Service level administrator

Service level administrator is the user role introduced by workload management whose main task is to ensure that overall installation operation is consistent with performance goals and objectives. The service level administrator is responsible for defining the installation's performance goals based on business needs and current performance.

Because the service definition terminology is similar to the terminology found in a Service Level Agreement (SLA), a service level administrator can communicate with the installation user community, with upper level management, and with MVS using the same terminology. When the service level requirements change, the service level administrator can adjust the corresponding workload management terms, without having to convert them into low-level MVS parameters.

1.11 WLM types of goals

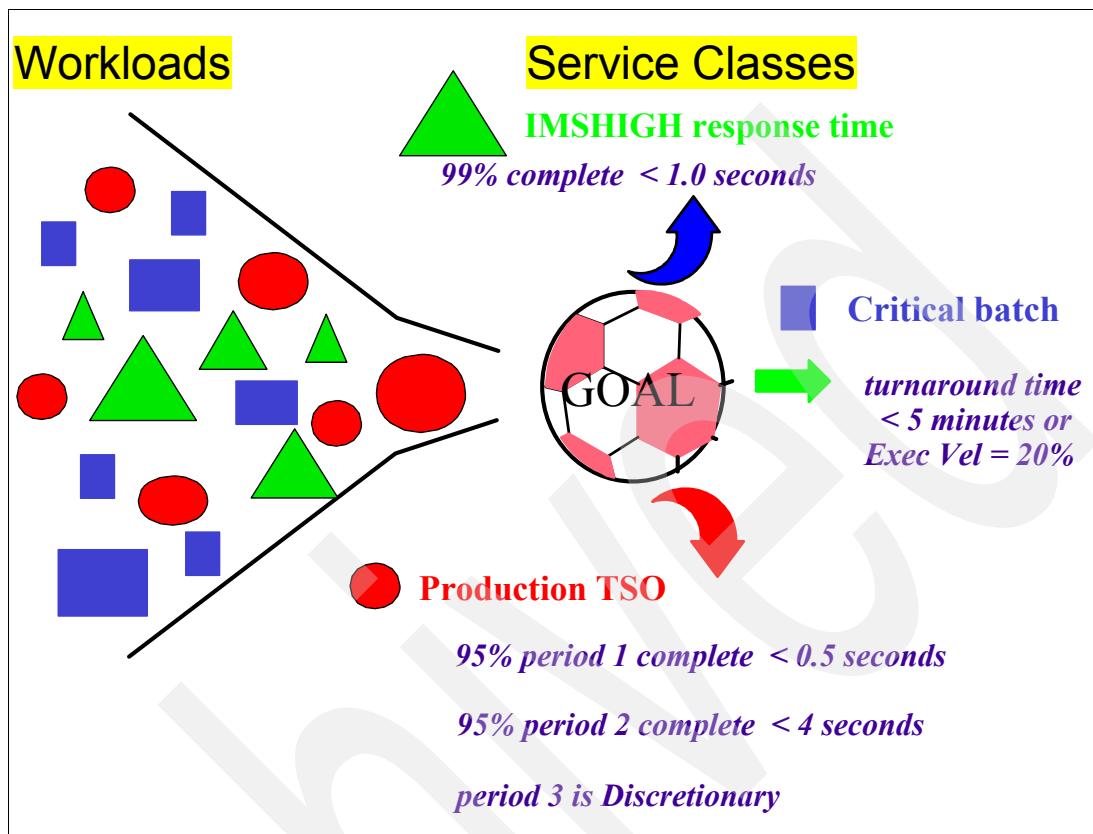


Figure 1-11 WLM types of goals

Figure 1-11 displays various types of WLM goals. These are discussed in more detail in the following sections.

Average response time goal

Average response time is the expected amount of time required to complete the transaction (within the z/OS host). This goal is submitted under the service class, in milliseconds, seconds, minutes, and hours. Note that this *only* includes the time spent on z/OS (for example, it does not include network time or the transaction time of other servers).

Average response time should be used when enclaves or address spaces produce a reasonable rate of ended transactions. Some examples could be TSO, Batch, CICS, IMS, OMVS, APPC, WebSphere® MQ, DB2 parallel query, DB2 stored procedure, DDF, and HTTP.

Note: Do not use this type of goal for address spaces where the transaction is the address space itself, or where there is no ending transaction, such as with DFSMSshm, RACF, and DFSORT.

For statistical reasons, transactions that are appropriate for an average response time goal should have at least 10 transaction completions within 20 minutes. If less, it might be difficult for WLM to react correctly because a single transaction might have a significant impact and affect the calculated response time. WLM is more efficient when it can compare a statistically valuable number based on multiple ended transactions with the defined goal.

Percentile response time

As you will see later, average response time can be influenced by a very few, long-running (outlier) transactions. To become more independent of such deviations from the average, you need a more stable goal definition. For example, suppose you want a certain number of transaction requests to end within a predefined time. Other transaction requests may run much longer, but the goal is considered to be achieved when the defined number of requests ends in the expected time period. This goal is known as a *percentile response time* goal and it defines the percentage of all transaction requests that must end within a predefined time limit.

Using time as a measure to manage the transaction on a system requires the system to have the ability to capture it. z/OS provides processes (dispatchable units as tasks and service requests), enclaves, and address spaces to execute the application programs. Sometimes these constructs are started once and live for a very long time. This means that WLM needs assistance from the middleware (such as CICS, IMS DC) to capture the response time. WLM also supports an infrastructure that allows the middleware to identify individual transaction requests, capture their response times, and thus allow WLM to manage them in their own service classes. This means that on z/OS, WLM is able to manage the service classes of transactions that belong to the same application and that are executed by a set of server address spaces.

To summarize, the percentile response time goal is the percentage of transactions in that period that should complete within the response time (for example, 80% of transactions ended in 0.5 of a second). If you have a problematic response time distribution towards the average, using a percentile goal is more useful than using an average goal. For example, nine transactions with a response time of 0.5 seconds and one transaction with a response time of 3 seconds have an average response time of 0.75, but 90% of them have a response time of 0.5 seconds.

Note: Workload Manager does not delay or limit transactions to achieve the response time goal when extra processing capacity exists, unless you are explicitly using a WLM capping function.

Execution velocity% goal

Refer to 1.13, “Execution velocity - discretionary goals” on page 25 for detailed information about this topic.

Discretionary goal

Refer to 1.14, “Discretionary goals” on page 27 for detailed information about this topic.

Service class periods

As shown in Figure 1-11 on page 21, a service class can contain multiple service class performance periods for transactions as they consume more and more resources. You can specify up to eight periods. The idea is to privilege the trivial transaction (the 80% that consumes 20% of the resources) with a first period difficult goal and a high importance. The ones that consume more than the DUR service units defined in the period are downgraded to the second period with an easier goal and less importance.

Then, when defining a service class period you specify a goal, an importance, and a duration for a performance period. The duration is the amount of service that period should consume before going on the next goal. The duration is specified in service units, as shown in Figure 1-12 on page 23.

1.12 Service units

- ❑ WLM/SRM monitors the dynamic performance characteristics of all address spaces under its control to ensure distribution of system resources as intended by the installation
- ❑ Performance characteristics are the rate at which an address space is receiving service relative to other address spaces competing for resources within the same domain
- ❑ The amount of service consumed by an address space is computed by the following formula:

```
service = (CPU x CPU Service Units)
          + (SRB x SRB Service Units)
          + (IOC x I/O Service Units)
          + (MSO x Storage Service Units)
```

Figure 1-12 Service units

Service units

The amount of system resources an address space or enclave consumes is measured in service units. Service units are calculated based on the CPU, SRB, I/O, and storage (MSO) service that an address space consumes.

Service units are the basis for period switching within a service class that has multiple periods. The duration of a service class period is specified in terms of service units. When an address space or enclave running in the service class period has consumed the amount of service specified by the duration, workload management moves it to the next period. The work is managed to the goal and importance of the new period.

Because not all kinds of services are equal in every installation, you can assign additional weight to one kind of service over another. This weight is called a *service coefficient*.

System Resources Manager

System Resources Manager (SRM) is a component of the system control program. It determines which address spaces, of all address spaces, should be given access to system resources and the rate at which each address space is allowed to consume these resources.

Service unit calculation

A *service unit* is the amount of service consumed by a work request as calculated by service definition coefficients and CPU, SRB, I/O, and storage service units.

One of the basic functions of WLM/SRM is to monitor the dynamic performance characteristics of all address spaces under its control to ensure distribution of system resources as intended by the installation.

A fundamental aspect of these performance characteristics is the rate at which an address space is receiving service relative to other address spaces competing for resources within the same domain.

The amount of service consumed by an address space is computed by the following formula:

```
service = (CPU x CPU Service Units)
        + (SRB x SRB Service Units)
        + (IOC x I/O Service Units)
        + (MSO x Storage Service Units)
```

CPU, SRB, IOC, and MSO are installation-defined service definition coefficients, as follows:

- ▶ CPU Service Units = task (TCB) execution time, multiplied by an SRM constant that is CPU model-dependent. Included in the execution time is the time used by the address space while executing in cross-memory mode (that is, during either secondary addressing mode or a cross-memory call). This execution time is not counted for the address space that is the target of the cross-memory reference.
- ▶ SRB Service Units = service request block (SRB) execution time for both local and global SRBs, multiplied by an SRM constant that is CPU model-dependent. Included in the execution time is the time used by the address space while executing in cross-memory mode (that is, during either secondary addressing mode or a cross-memory call). This execution time is not counted for the address space that is the target of the cross-memory reference.
- ▶ I/O Service Units = measurement of individual data set I/O activity and JES spool reads and writes for all data sets associated with the address space. SRM calculates I/O service using either I/O block (EXCP) counts or device connect time (DCTI), as specified on the IOSRVC keyword in the IEAIPSxx parmlib member.

Note that if DCTI is used to calculate I/O service, then operations to VIO data sets and to devices that the channel measurement facility does not time are *not* included in the I/O service total. When an address space executes in cross-memory mode (that is, during either secondary addressing mode or a cross memory call), the EXCP counts or the DCTI will be included in the I/O service total. This I/O service is not counted for the address space that is the target of the cross-memory reference.

- ▶ Storage Service Units = (central page frames) x (CPU service units) x 1/50, where 1/50 is a scaling factor designed to bring the storage service component in line with the CPU component.

Note that the main storage page frames used by an address space while referencing the private virtual storage of another address space through a cross-service are *not* included in the storage service unit calculation. These frames are counted for the address space whose virtual storage is being referenced.

1.13 Execution velocity - discretionary goals

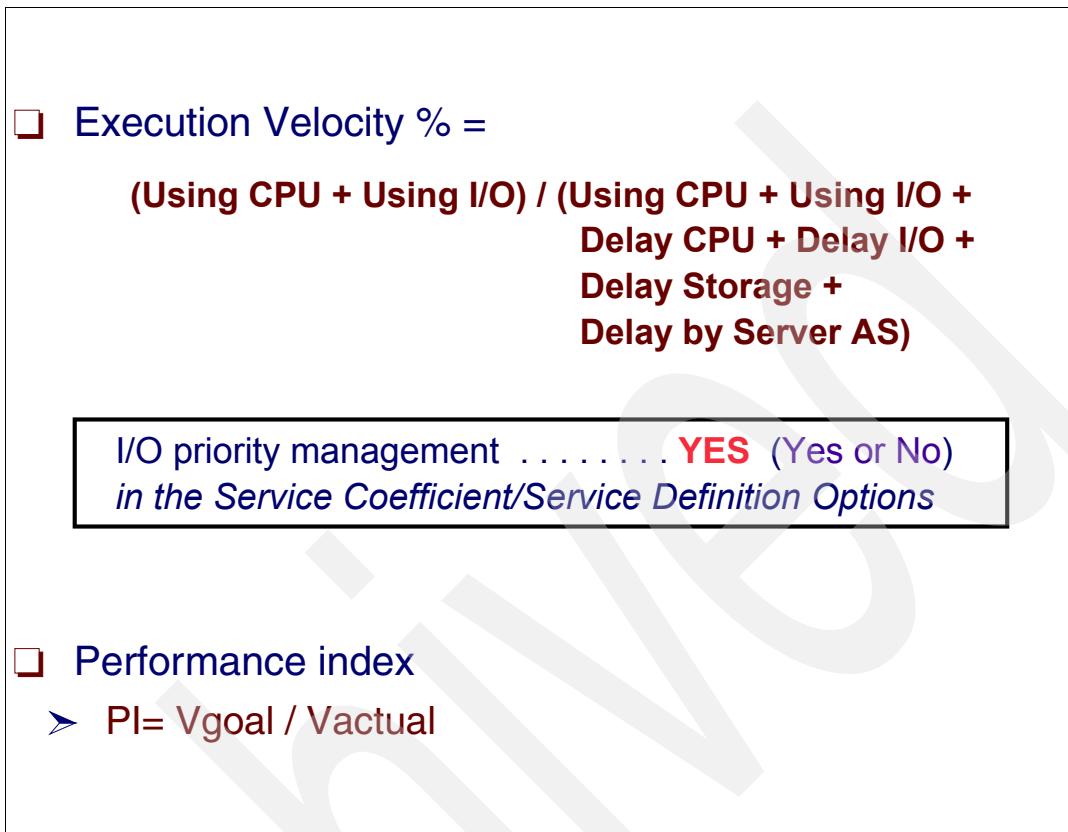


Figure 1-13 Execution velocity and discretionary goals

Execution velocity goal

Many address spaces provide services in the system that cannot be easily attributed to a certain type of transaction. For those address spaces, it is also necessary to understand how fast they progress in the system and define a goal for them. Because WLM is not able to capture the response time of this type of transaction, it must use the information about the Delay state and Using state values that it observes when the transaction tries to use resources in the system. *Delay state* is a sampled figure of the transaction wait-for-resource time. *Using state* is a sample figure of the transaction resource service time.

Execution velocity goals define how fast work should run when ready, without being delayed for processor, storage, I/O access, and address space queue delay. Execution velocity goals are intended for work for which response time goals are not appropriate, such as started tasks or long-running batch work. Figure 1-13 illustrates execution velocity and discretionary goals.

Execution velocity

The speed in the system can now be expressed by the acceptable amount of delay for the transaction when it executes. The speed is known as *execution velocity*, which is a number between 0 and 100. 0 means that the transaction was completely delayed over the measurement interval. 100 means that all measured samples are using samples and that the transaction did not encounter any delays for resources managed by WLM. Generally we may say:

$$\text{EXEC VELOCITY\%} = \text{USING SAMPLES} / (\text{USING SAMPLES} + \text{DELAY SAMPLES}) \times 100$$

Where:

- ▶ USING SAMPLES include all types of PUs (CPU, zAAP, zIIP) using samples and I/O using samples.
- ▶ DELAY SAMPLES include all types of PUs delays, I/O delays, storage delays, and servant address space queue delays.

To include the I/O figures in both Using and Delay samples in the Execution Velocity goal formula, you need the option I/O Management to be set to YES in the service policy.

Processor delays also include processor capping delays, which come from resource group capping. In this case WLM cannot do anything to avoid such delays because they are directly caused by capping, as required by the installation. Storage delays consists of a variety of different delay reasons (such as page faulting and WLM address space swapping). Queue delays can either be JES queue delays (for batch job transaction) or address spaces queue delays for applications like WebSphere or DB stored procedures, which exploit the WLM queuing services.

Understanding velocity goal

Next, we compare the potential goals that can be defined for transactions in the system. On one hand, we can have a response time goal; independently of whether we use an average or percentile response time goal, we can always immediately understand what this means with respect to how fast the transaction progresses in the system.

On the other hand, we have an execution velocity goal. We see that an execution velocity of 100% means no delay (the goals managed by WLM as CPU, I/O, and storage), and therefore it translates into maximal speed. But what does an execution velocity goal of 30% mean?

If you want to define a reliable goal that can be achieved under all circumstances, then you have to understand how the system behaves during rush hour. Such goals may be easily achieved during other times because there is no contention and everything can progress very quickly.

Do not define an execution velocity goal higher than 85%. It will be a very difficult goal to achieve and will cause worthless WLM attention without a clear profit.

When there are two service classes for different transactions, one for which you can measure the response time and one for which you cannot measure the response time, you need a way to compare the potential goal settings against each other. Because it is very difficult to understand how an execution velocity translates into a response time, you have to capture the execution velocity of the transaction being managed towards response time goals simply because you need to understand what this means as an execution velocity. A way for normalizing these different type of goals is through the calculation of a performance index (PI), a number that indicates (and also indicates to WLM) how far the service class period is from the goal, not considering the type of the goal.

1.14 Discretionary goals

- ❑ Used for work that does not need a specific performance goal
- ❑ Goals for low priority work
 - Mainly batch
- ❑ Run work only when there is capacity available
- ❑ Discretionary goal considerations
 - Discretionary is always the goal of the SYSOTHER service classes.
 - Only allowed in last period of a service class
 - Has fixed I/O priority of 248
 - (PI) is always equal to 0.81

Figure 1-14 Discretionary goals

Discretionary goals

With discretionary goals, as listed in Figure 1-14, WLM decides how best to run this work. Because the prime WLM responsibility is to match resources to work, discretionary goals are best used for work for which you do not have a specific performance goal. For a service class with multiple performance periods, you can specify discretionary only as the goal in the last performance period.

Discretionary work is run using any system resources not required to meet the goals of other work. If certain types of other work are overachieving their goals, that work may be “capped” so that the resources may be diverted to run discretionary work.

Discretionary goals are for low priority work for which you do not have any particular performance goal. Workload management then processes the work using resources not required to meet the goals of other service classes.

In addition to setting response time or velocity goals, you might want to have a transaction running in the system for which no concrete goal must be achieved. This transaction should only run when the transactions with a specific goal are satisfied. This transaction is classified to service classes with a *discretionary* goal and thus tells the system that it should only run when service classes with higher importance do not need the resources to achieve their goals.

Discretionary goal considerations

Remember the following when setting up your service definitions:

- ▶ Discretionary is always the goal of the SYSOTHER service class, which is the default for all subsystems (excluding STC) in the classification rules.
- ▶ Discretionary is only allowed in *last* period of a service class. It does not make sense to have a discretionary goal in a period and a different type of goal in the next period.
- ▶ Discretionary has a fixed I/O priority of 248, which is the minimum value used by WLM.
- ▶ The Performance index (PI) is always equal to 0.81. There is no reason for varying the PI if the service class does not have a numeric goal. It is key to be less than one, to indicate that everything is okay. The 0.81 PI can be used as an eye-catcher, in some reports.

1.15 Assigning service classes

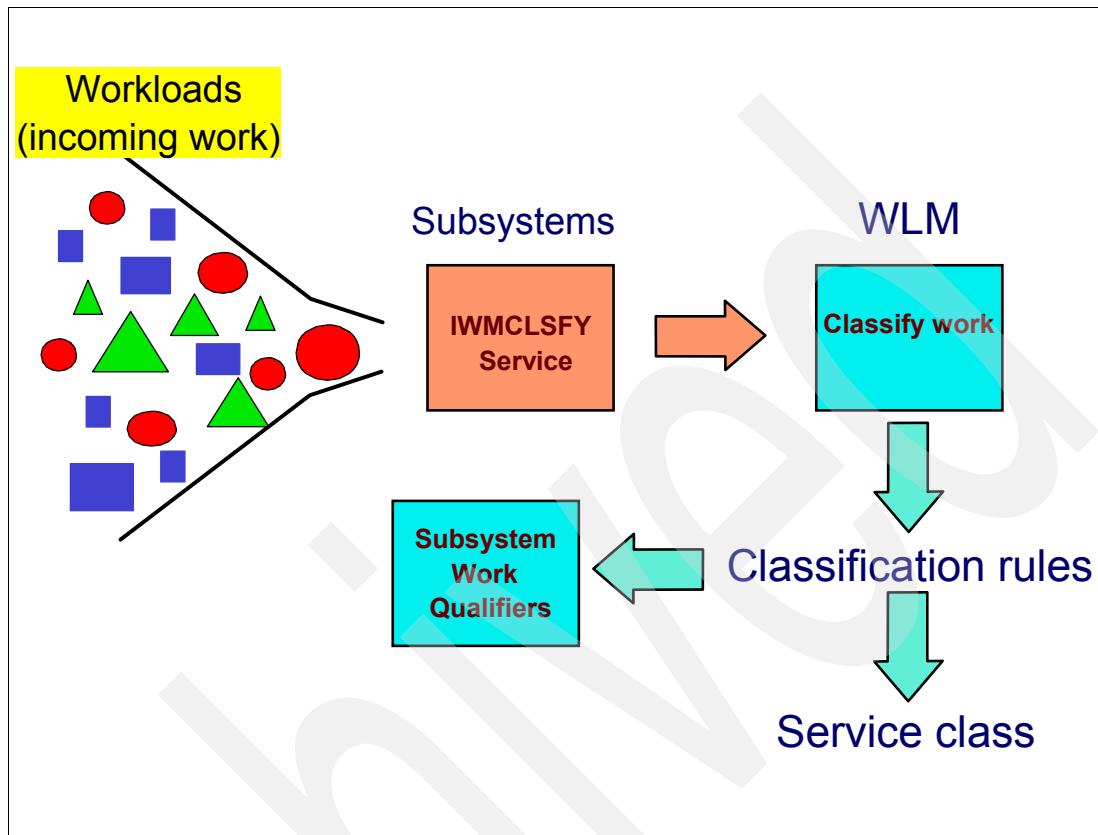


Figure 1-15 Assigning service classes to incoming work using classification rules

IWMCLSFY service

When each subsystem's transaction manager receives a work request, it should issue the IWMCLSFY service to associate an incoming work request with a service class. WLM receives this classification request and then uses the classification rules defined by the installation to classify the new work by assigning a service class. Every piece of work that enters the system will have a service class assigned to it via the IWMCLSFY service. Figure 1-15 illustrates assigning service classes to incoming work using classification rules.

Classification rules

Classification rules are the rules that WLM uses to associate a performance goal or a resource group with work by associating incoming work with a service class. Optionally, classification rules can also associate incoming work with a report class, similar to a report performance group. These classification rules, defined by an installation, are the rules you define to categorize work into service classes, and optionally report classes, based on work qualifiers.

Subsystem work qualifiers

A work qualifier is what identifies a work request to the system. The first qualifier is the subsystem type that receives the work request. There is one set of classification rules in the service definition for a sysplex. These classification rules are defined by the installation. They are the same regardless of what service policy is in effect; a policy cannot override

classification rules. Define classification rules after you have defined service classes, and ensure that every service class has a corresponding rule.

The classification rules for a subsystem are specified in terms of transaction qualifiers such as job name or transaction class. These qualifiers identify groups of transactions that should have the same performance goals and importance.

Work qualifiers

Each subsystem has its own transaction work qualifiers. The attributes of incoming work are compared to these qualifiers and, if there is a match, the rule is used to assign a service class to the work. A subsystem can also have a default class for work that does not match any of the rules.

Note: Not all work qualifiers are valid for every subsystem type; they are subsystem-dependent.

1.16 Work qualifiers

AI	Accounting Information	PNG	Plan Name Group
CI	Correlation Information	PR	Procedure Name
CN	Collection Name	PRI	Priority
CT	Connection Type	PX	Sysplex Name
CTG	Connection Type Group	SE	Scheduling Environment
ESC	EWLM service class name	SI	Subsystem Instance
ETC	EWLM transaction class name	SIG	Subsystem Instance Group
LU	LU Name	SPM	Subsystem Parameter
LUG	LU Name Group	SSC	Subsystem Collection
NET	Net ID	SY	Sysname
NETG	Net ID Group	SYG	Sysname Group
PC	Process Name	TC	Transaction Class
PF	Perform	TCG	Transaction Class Group
PFG	Perform Group	TN	Transaction Name
PK	Package Name	TNG	Transaction Name Group
PKG	Package Name Group	UI	Userid
PN	Plan Name	UIG	Userid Group

Each subsystem uses its own set of qualifiers

Figure 1-16 WLM work qualifiers listed by their abbreviations

WLM work qualifiers

When a subsystem receives a work request, the transaction manager of each subsystem should pass work qualifiers using the IWMCLSFY service to WLM. Each installation defines the classification rules in the workload management ISPF application. Each transaction manager does not support all of the qualifier types, so each subsystem passes its supported work qualifiers to the WLM ISPF application. These work qualifiers are used by WLM to classify that element of work into a service class. Figure 1-16 displays WLM work qualifiers listed by their abbreviations.

Work qualifier groups

Groups are available for grouping together work qualifiers to make classification simpler. You can create groups to collect together work when you do not have a standard naming convention that allows masking or wildcarding. A *group* is a collection of the same work qualifiers. For example, you may want to create a group of started tasks because you want to assign them all to the same service class.

Note: For many of these qualifiers, you can specify classification groups by adding a G to the type abbreviation; see Figure 1-16.

1.17 Service classes

```
Line 00000000 Col 001 072
SCROLL ==> PAGE
***** Top of Data *****
* Service Class BATCHHI - Batch Workload, above BATCH

Created by user HAIMO on 2004/06/25 at 15:11:05
Base last updated by user HAIMO on 2004/06/25 at 15:12:09

Base goal:
CPU Critical flag: NO

# Duration Imp Goal description
- ----- -
1 3 Execution velocity of 60

***** Bottom of Data *****
```

Figure 1-17 Service class construct

Service classes

A *service class* is a named group of work within a workload with the following similar performance characteristics:

- ▶ Performance goals
- ▶ Resource requirements
- ▶ Business importance to the installation

WLM manages a service class period as a single entity when allocating resources to meet performance goals. A service class can be associated with only one workload. You can define up to 100 service classes.

Service class construct

The service class (SC) describes a group of transactions within a workload with similar performance characteristics; see Figure 1-17. A service class is associated with only *one* workload, and it can consist of one or more periods (as specified through the DURATION keyword).

Service class period parameters

Because some work has variable resource requirements, WLM provides performance periods where you specify a series of varying goals and importances. You can define up to eight performance periods for each service class.

A service class period has the following parameters:

- ▶ Importance

The *importance* tells how important the goal is to be honored by WLM. If different service class periods are achieving their goals, the importance is used as a tie breaker to select the service class to be helped first.

- ▶ Performance goal type

Figure 1-17 on page 32 shows an example of a service class (BATCHHI) with just one period with an Execution Velocity% goal of 60% and an importance of 3.

1.18 Service class period

Figure 1-18 Service class with two periods

Service class period

Performance periods are available for work that has variable resource requirements and for which your goals change as the work uses more resources. You specify a goal, an importance, and a duration for a performance period. Duration is the amount of service that period should consume before going on the next goal. Duration is specified in service units.

Note: You can define multiple performance periods for work in subsystems which use address spaces or enclaves to represent individual work requests. Multiple periods are not supported for work in the IMS and CICS subsystem work environments because service units are accumulated to the address space, not the individual transactions. So, the system cannot track a duration for those transactions.

Sometimes the installation knows in advance that the transactions coming from one department are trivial. In such cases you simply assign to them a service class with a difficult and very important goal. This translates (by WLM) into high priorities for the dispatchable units executing the programs of such transactions. However, when the same department mixes trivial and non-trivial transactions, you may use the concept of service class periods to protect the trivial ones.

As an example, Figure 1-18 shows one service class (DDF) with two periods. It works like this:

- ▶ All transactions for this service class start in the first period enjoying high priorities because of the difficult goal and the importance of 2.
- ▶ If a transaction ends before consuming less than 1,000 service units (includes CPU, SRB, I/O, and storage service units) it is considered trivial.
- ▶ If the transaction goes beyond such a mark, it is considered non-trivial and migrates to the second period where the goal is easier (discretionary, in this case) and the priorities lower.

The value of duration is determined empirically by watching RMF data about the percentage of transactions ending in the first period. If this is more than 80%, then the duration must be decreased. If this is less than 80%, then the duration must be increased.

Optional service class definitions

The following parameters are optional when defining service classes:

- ▶ Resource Group (optional)
- ▶ CPU Critical (optional)

This can be defined for a service class for long-term CPU protection for critical transactions.

- ▶ Storage Critical (optional)

This can be defined for a service class to assign long-term storage protection to critical transactions.

1.19 Subsystems supported by WLM

<input type="checkbox"/> ASCH	<input type="checkbox"/> LSFM
<input type="checkbox"/> CB	<input type="checkbox"/> MQ
<input type="checkbox"/> CICS	<input type="checkbox"/> NETV
<input type="checkbox"/> DB2	<input type="checkbox"/> OMVS
<input type="checkbox"/> DDF	<input type="checkbox"/> SOM
<input type="checkbox"/> EWLM	<input type="checkbox"/> STC
<input type="checkbox"/> IMS	<input type="checkbox"/> TCP
<input type="checkbox"/> IWEB	<input type="checkbox"/> TSO
<input type="checkbox"/> JES	<input type="checkbox"/> SYSH

Figure 1-19 WLM supported subsystems

Defined subsystem types

The following subsystem types are supported by WLM, as illustrated in Figure 1-19. These are the subsystems that can be defined in the WLM policy where you define goals, assign service classes using the classification rules, and assign a performance index to control the importance of that particular work running in the system.

- ASCH** The work requests include all APPC transaction programs scheduled by the IBM-supplied APPC/MVS transaction scheduler.
- CB** The work requests include all Component Broker client object method requests.
- CICS** The work requests include all transactions processed by CICS Version 4 and higher.
- DB2** The work requests include only the queries that DB2 has created by splitting a single, larger query and distributed to remote systems in a sysplex. The local piece of a split query, and any other DB2 work, is classified according to the subsystem type of the originator of the request (for example, DDF, TSO, or JES).
- DDF** The work requests include all DB2 distributed data facility (DB2 Version 4 and higher) work requests.
- EWLM** This allows you to assign EWLM transaction and service classes by name to WLM service classes.
- IMS** The work requests include all messages processed by IMS Version 5 and higher.

IWEB	The work requests include all requests from the World Wide Web being serviced by the Internet Connection Server (ICS), Domino® Go Webserver, or IBM HTTP Server for z/OS. These requests also include those handled by the Secure Sockets Layer (SSL). Also included are transactions handled by the Fast Response Cache Accelerator.
JES	The work requests include all jobs that JES2 or JES3 initiates.
LSFM	The work requests include all work from LAN Server for MVS.
MQ	The work requests include MQSeries® Workflow work such as new client server requests, activity executions, activity responses, and subprocess requests.
NETV	The work requests include NetView® network management subtasks and system automation (SA) subtasks created by Tivoli® NetView for z/OS.
OMVS	The work requests include work processed in z/OS UNIX System Services forked children address spaces. (Work that comes from an enclave is managed to the goals of the originating subsystem.)
SOM	The work requests include all SOM client object class binding requests.
STC	The work requests include all work initiated by the START and MOUNT commands. STC also includes system component address spaces such as the TRACE and PC/AUTH address spaces.
TCP	The work requests include work processed by the z/OS Communications Server.
TSO	The work requests include all commands issued from foreground TSO sessions.
SYSH	This identifies non-z/OS partitions (for example, a Linux® partition) in the LPAR cluster that need to be managed by WLM according to business goals set for the partition.

Archived

WLM ISPF application

This chapter describes the WLM online panel-based application used for setting up and adjusting the service definition. You specify the service definition through this ISPF administrative application. Service policies are activated by an operator command, or through the ISPF administrative application utility function.

A service definition contains all of the information necessary for workload management processing. Based on this information, you should be able to set up a preliminary service definition using the worksheets provided. Then, you can enter your service definition into the ISPF administrative application with ease.

2.1 ISPF administrative application

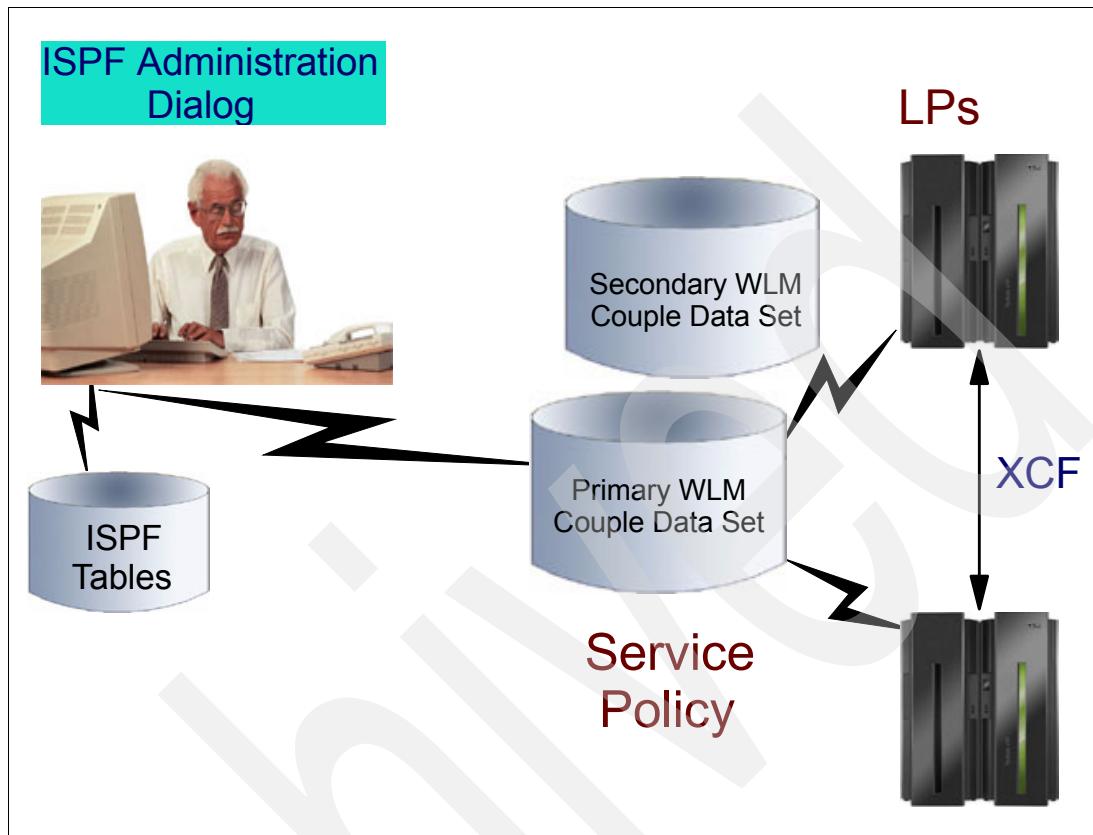


Figure 2-1 Setting up a service policy

Service definition

The service definition (SD) is used to express your installation's business goals to your sysplex. The SD must be installed on a WLM couple data set, and a service policy (contained in this SD) activated. Only service policies in the SD installed on the WLM couple data set can be activated. Figure 2-1 illustrates setting up a service policy. A WLM couple data set has an automatic backup. You may define a minimum of one policy in a service definition. A service definition contains policies, workloads, service classes, classification rules, resource group (optional), application environment (optional), and scheduling environment (optional.).

ISPF tables

The WLM SD is stored in ISPF tables, or in XML format. When a new release adds certain specifications to the SD, changes to the ISPF tables are required. When you open an SD that has a table structure different (older) from the one currently used by the WLM application, the WLM application automatically updates the SD structure. After this occurs, the SD cannot be read by older levels of the WLM application unless the compatibility APARs are installed.

WLM couple data sets

When you set up your SD, you identify the workloads, the resource groups, the service classes, the service class periods, and goals based on your performance objectives. Then you define classification rules and one or more service policies. This information makes up the base SD. This SD is written to the couple data sets by WLM and finally you activate one of the service policies from the SD.

2.2 Defining a service definition

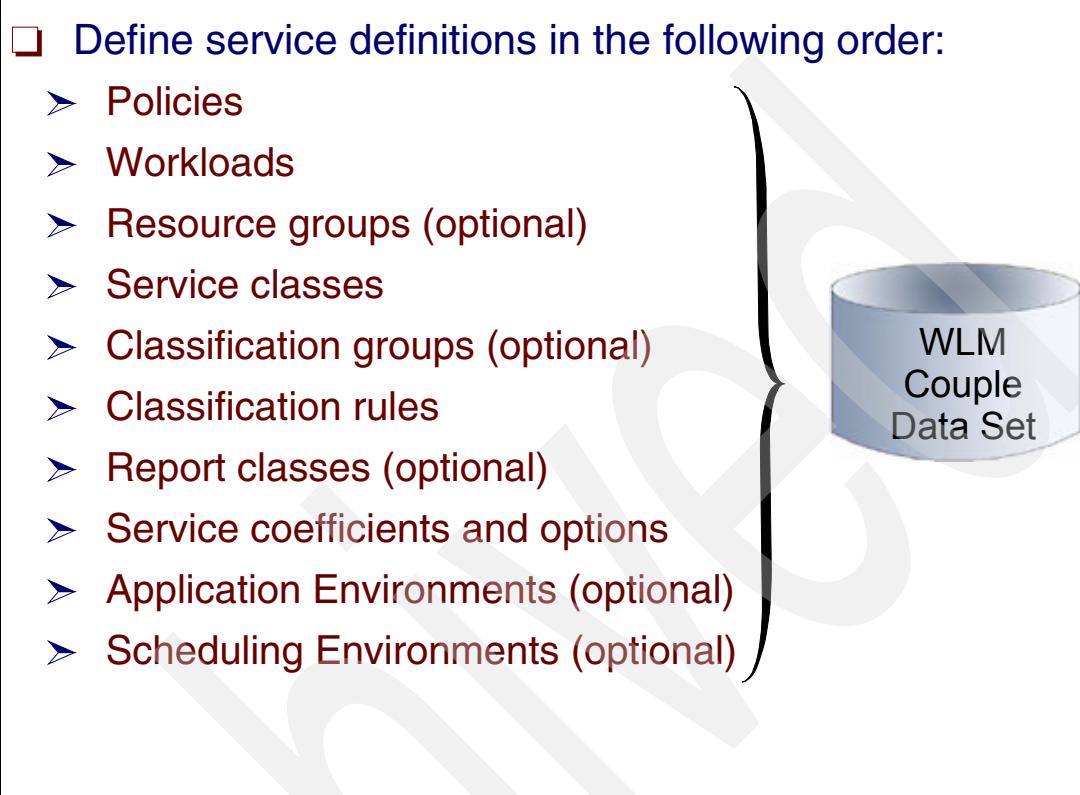


Figure 2-2 Service definition options

Definition options

When you define your service definition for the first time, define it in the following order:

- ▶ **Policies** - A policy consists of a name, a description, and policy overrides. The first time you set up a service definition, define a policy name and description. If you do not have a business need to change your goals, you can run with one service policy, without any policy overrides. All the new service policies must share the same set of service classes and workloads. The difference is the numeric values of the goals. Then, when you add a new policy, you override the original; refer to Figure 2-3 on page 43. You use a policy override only if you have a business need to change a goal for a certain time, such as for the weekend or nighttime. You can define your policy overrides after you have defined your service classes.
- ▶ **Workloads** - A workload logically consists of a group of one or more service classes. You associate a workload with a service class in the Service Class panel. The reason for having workloads is for introducing a unit of account. Enter your workloads before creating your service classes.
- ▶ **Resource groups (optional)** - A resource group is a minimum or maximum amount of processing capacity. You associate a resource group with a service class in the Service Class panel. Enter resource groups before creating your service classes.
- ▶ **Service classes** - A service class is a group of transactions with similar performance goals, resource requirements, or business importance. You make the association with a workload and a resource group in the service class panel. You associate a service class

with incoming work in the classification rules panel. Enter service classes before creating classification rules. After you have created a service class, you can create a policy override. You specify the policy override by selecting Service Policies from the Definition Menu, and then specifying the action code for Override service class or Override resource group.

- ▶ **Classification groups** (optional) - You use groups to simplify classification. You associate a classification group with a service class in the classification rules panel. If you intend to use them, create groups before creating classification rules.
- ▶ **Classification rules** - Classification rules assign incoming transactions to service classes. Before you create your classification rules, you must understand which subsystem's work is represented in each of your service classes. This selection list is primed with all of the IBM-Supplied subsystem types. They are reserved names.
- ▶ **Report classes** (optional) - A report class is a group of transactions for which you want reporting data. You do not have to define report classes before assigning them to transactions in classification rules. You can create them from within the classification rules menu. Just for comparison, a service class is a unit for goals and for accounting, a report class is a unit for accounting.
- ▶ **Service coefficients**/options - Service coefficients define the weight to be applied to one type of service units over another (CPU, I/O, and storage) in the calculation of service rates. You can enter new values for the CPU, IOC, MSO, and SRB service coefficients, although the recommendation is to leave what is set by the IBM installation pack.
- ▶ **Application environment** - An application environment is a set of identical application transactions that are executed in a set of identical server address spaces. These address spaces are started with the same JCL procedure and can access exactly the same resources. You can have workload management, depending on the arrival demand, start and stop these server address spaces automatically, or do this manually or through automation. You define the application environment, an optional procedure name for starting the server address spaces, and any start parameters needed for the start procedure.
- ▶ **Scheduling environment** - A scheduling environment is a list of resource names along with their required states. By associating incoming jobs with a scheduling environment, you ensure that a job is assigned to a system only if that system satisfies all of the requirements. You define the scheduling environment, listing all of the resource names and required states that are contained within. You also define the resource names themselves.

2.3 Service policy override

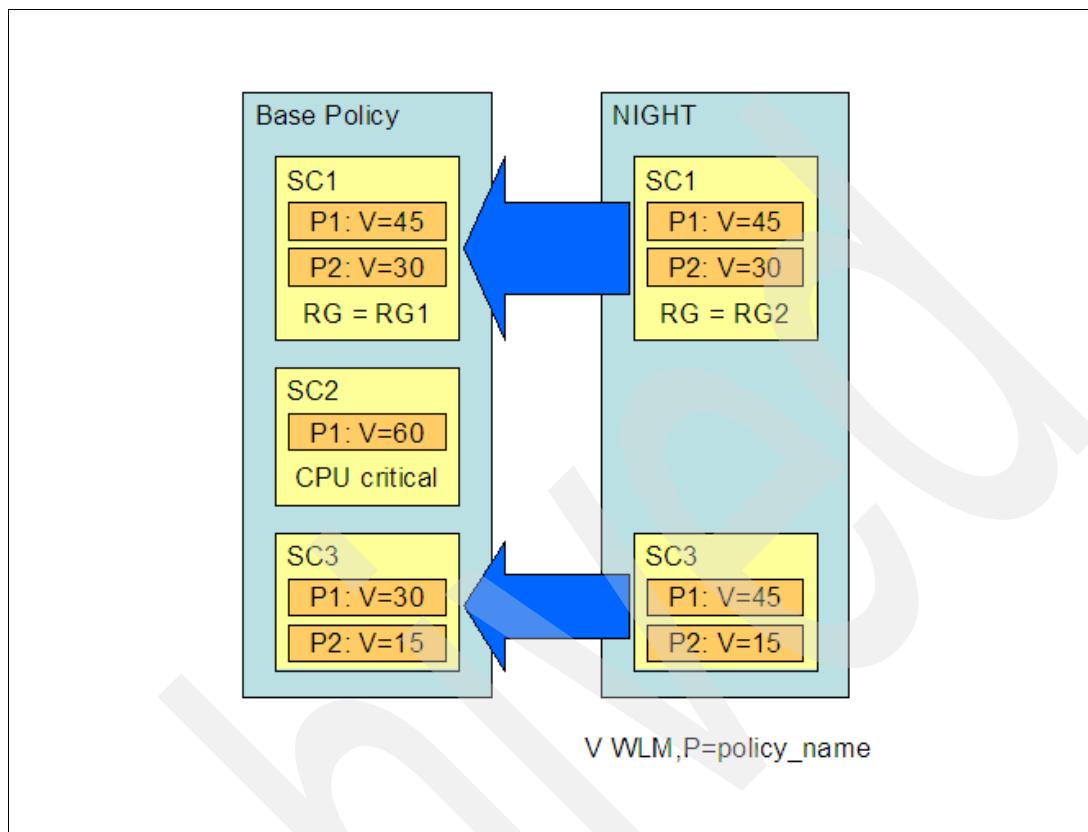


Figure 2-3 Service policy override

Service policy override

You define all service policy overrides from the service policy selection list. You can define three kinds of service policy overrides:

- ▶ Override service class goals
- ▶ Override resource group assignment
- ▶ Override resource group attributes

To override a service class goal, choose either the action code or the menu bar option to Override Service Classes. Figure 2-4 shows a selection list where you have chosen to override the service classes for the weekend policy.

Service-Policy View Notes Options Help					
Service Policy Selection List				ROW 1 TO 3	
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, 7=Override Service Classes, 8=Override Resource Groups, /=Menu Bar					
---Last Change---					
Action	Name	Description	User	Date	
—	HOLIDAY	Policy for shut-Down holidays	SALLA	2009/12/04	
—	WEEKDAY	Policy for Mon - Friday	SALLA	2009/12/04	
—	WEEKEND	Policy for Fri - Sun	SALLA	2009/12/04	
—	NIGHT	Policy for Night Shift	SALLA	2009/12/04	
***** BOTTOM OF DATA *****					
Command ==> _____					

Figure 2-4 Service Policy override panel

A service policy override enables an installation to apply different sets of goals under different sets of conditions (time of day, disaster scenarios, and so on) with no disruption of ongoing work.

Alternate policy

An alternate policy can be activated by the console command **V WLM,P=NIGHT**, which means that policy-switching can be automated. Policy changes take effect across the entire sysplex.

The base policy contains a set of service class period definitions. For each period, the definition includes an importance, a goal type, and a goal value. For the service class as a whole, the policy may also specify CPU-related constraints: “CPU Critical” and a resource group assignment (which may provide a maximum, a minimum, or both). Any or all of these may be overridden for any or all of the service classes.

In Figure 2-3 on page 43, the characteristics of SC1 and SC3 change, either on issuance of the **V WLM,P=NIGHT** console command or when IWMARIN0 is used to activate the NIGHT policy. In the overridden policy, SC1 gets a new resource group assignment but none of its period goals are changed. SC2 is not overridden at all. SC3's period goals change.

Only one override policy can be in effect at any time; you cannot partially override another override. An override cannot affect the classification rules, resource group values, or anything other than the attributes of a service class and its periods.

2.4 Creating a service definition

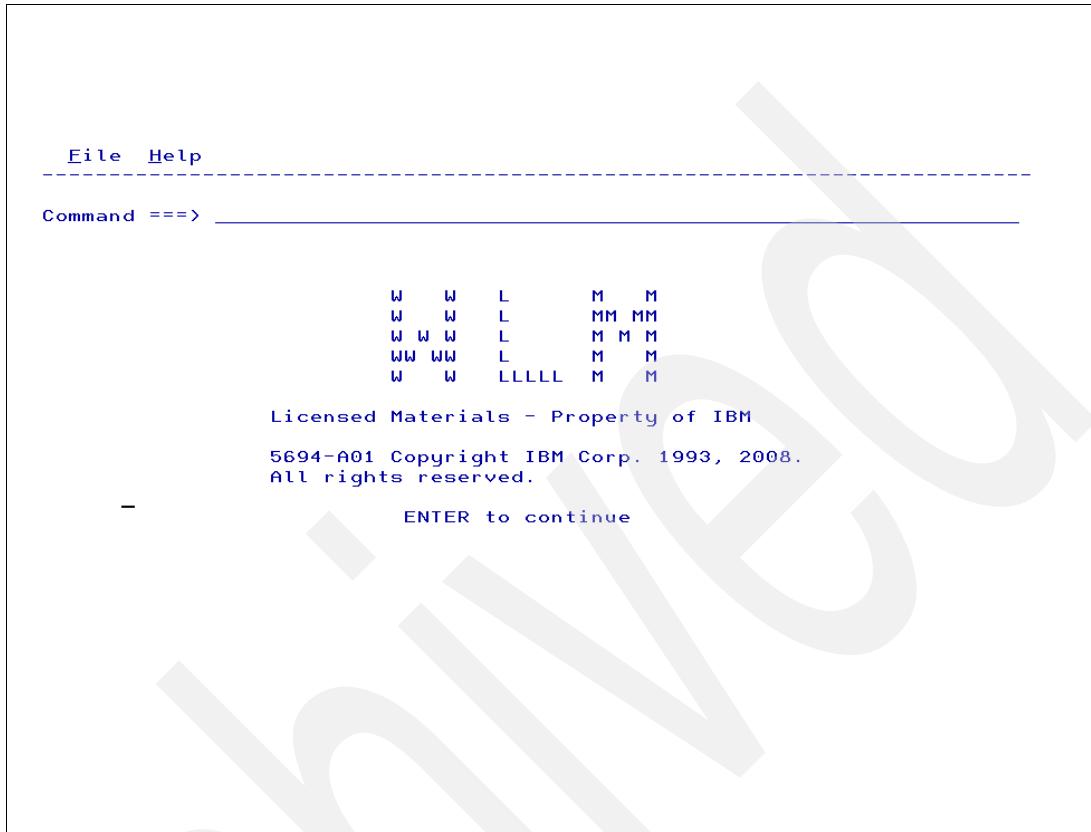


Figure 2-5 WLM - the first panel

Creating a service policy

A workload management ISPF application contains an installation's goals for work in a service definition, which is comprised of service policies. The performance monitors may access the service policy information and report on how well the installation is doing in processing towards the goals in the policy. The services report information based on the service classes defined in the service policy. They also provide delay information about work managed by subsystems using the execution delay monitoring services.

In the following sections we guide you through the WLM panels. These panels illustrate how to create a WLM policy. Figure 2-5 displays the first panel that is shown when you request the WLM application under ISPF.

Starting the ISPF application

After entering ISPF option 6, on the command line, to start the application, specify:

```
ex 'SYS1.SBLSCLI0(IWMARINO)'
```

Entering WLM

Note that when you have access to this panel, you can use the Help facility to clarify concepts as needed. To reach the next panel, press Enter. The full ISPF application is called IWMARINO.

2.5 Creating a service policy (1)

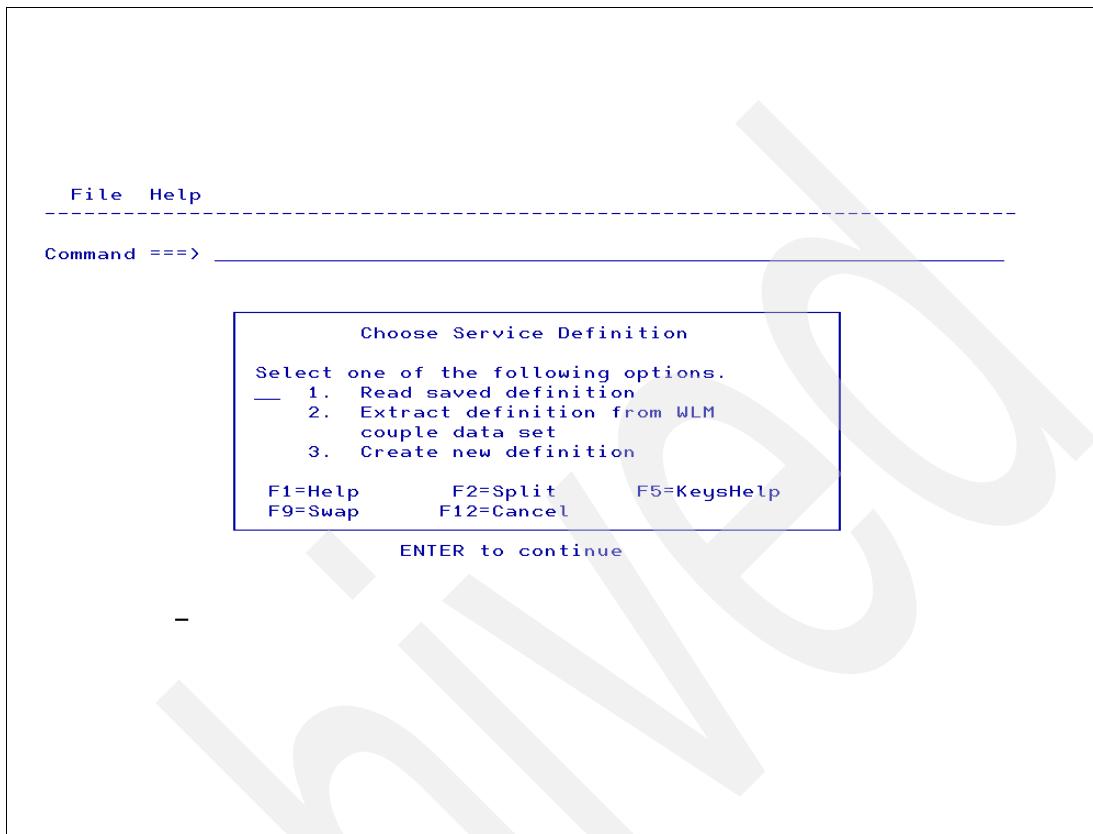


Figure 2-6 Service Definition panel

Service Definition panel

There are three options available to work with a service definition, as shown in Figure 2-6. You can work with one service definition at a time in the application. From the Service Definition panel you can:

- ▶ Create a new service definition in a PDS member (option 3).
With this option, you go to the Definition menu, where you define a service definition. When you exit the Definition menu, you go to the Specify disposition of the Service Definition panel. Here you can specify *one* of the following actions:
 - Save definition to a data set. This brings you to the Save to... panel where you specify the ISPF data set name to contain the service definition. You confirm the creation of the data set on the Create Data Set panel.
 - Install a definition on the WLM couple data set. This installs the service definition on the WLM couple data set. You must have previously allocated a WLM couple data set. There is no ISPF data set created when you choose this option.
 - Discard changes. This discards all changes made, and you exit from the application.
- ▶ Select Read saved definition to work with an existing service definition, which is stored in an ISPF data set (option 1).
With this option, you go to the Read Saved Definition menu where you can select the data set containing the service definition (option 2).
- ▶ Select Extract definition from WLM couple data set to extract the definition currently on the WLM couple data set.

2.6 Creating a service policy (2)

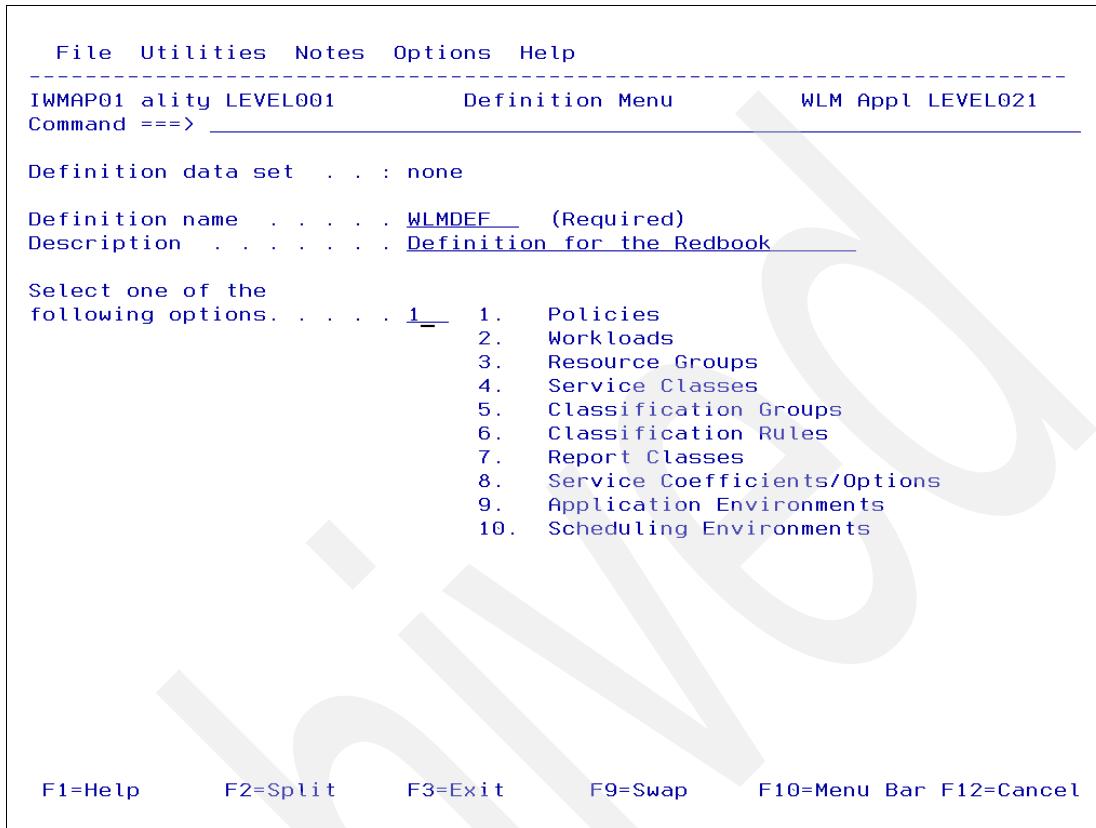


Figure 2-7 Definition menu

Definition menu

On the top left corner of the panel, as shown in Figure 2-7, notice the panel ID (IWMPA01, in this case). This name is presented as a consequence of the TSO/ISPF command PANELID.

This menu lists WLM constructs that may be assigned to define the service definition. The name of the service definition is a name you choose and you define some text that describes the service policy definition that is being created.

The Definition Menu is considered the home base panel. A significant amount of function is accessible from this panel, and you will find yourself returning to it frequently. One quick way to do that from just about anywhere in the application is to press F4=Return. On the definition menu header line appears the functionality level (top left side) and the WLM application level (top right side). The functionality LEVEL021 represents the highest level function in the current service definition. The WLM Appl LEVEL021 is the level of the WLM ISPF application.

Definition data set

The definition data set is the data set containing the service definition. The definition data set is the name of a PDS data set containing the service definition temporarily. When the service definition is in the WLM couple data set, it does not apply. Then, it is *none* in the following cases:

- ▶ When you extract a service definition. The definition data set is none until you save it into an MVS partitioned data set. You have the option to save an extracted service definition

into an MVS PDS, or install the definition on the WLM couple data set when you exit the Definition menu.

- ▶ When you create a new service definition. The definition data set is none until you save it into an MVS PDS. You have the option to save the service an extracted service definition into an MVS PDS, or install the definition on the WLM couple data set when you exit the Definition menu.
- ▶ When you create a new service definition. The definition data set is none until you save it into an MVS PDS. You have the option to save the service definition into an MVS PDS, or install the service definition on the WLM couple data set when you exit the Definition menu.

Definition name

Choose a name for the definition. Optionally, you can define a description for the policy.

Select option 1 to create a policy and press the Enter key. You will reach the Create a Service Policy panel shown in Figure 2-8 on page 49.

2.7 Creating a service policy (3)

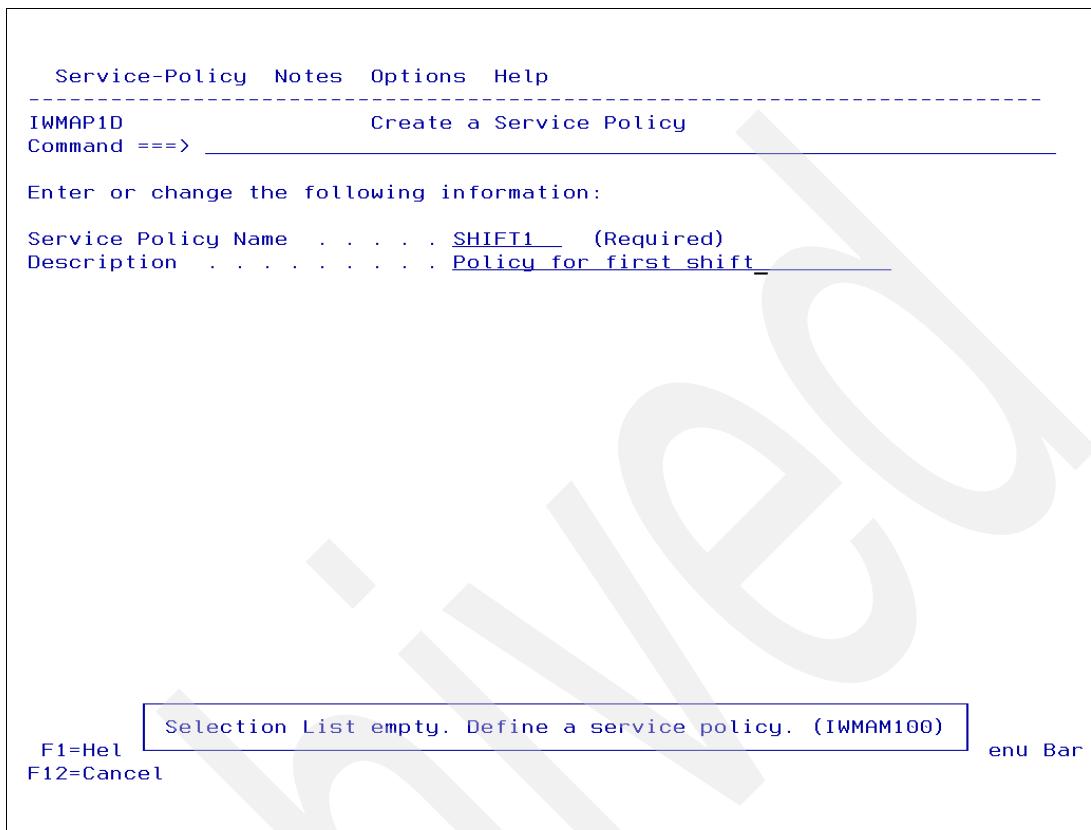


Figure 2-8 Define a policy

Define a policy

On this panel (IWMAP1D), you must type in the name you want to give your policy. Optionally, you can type in a description of the new service definition.

```
Service Policy Name . . . . . SHIFT1 (Required)
Description . . . . . . . . . Policy for first shift
```

Choose names that relate to the installation policies you need to create. The service policy reflects goals for work, expressed in terms commonly used in service level agreements (SLAs). Because the terms are similar to those commonly used in an SLA, you can communicate with users, with business partners, and with z/OS using the same terminology.

The service policy is a named set of overrides to the performance goals and processing capacity boundaries in the service definition. A policy applies to all of the work running in a sysplex. Because processing requirements change at different times, service level requirements may change at different moments. If you have performance goals that apply to different times or a business need to limit access to processor capacity at certain instants, you can define multiple policies. To start workload management processing, you must define at least one service policy. You can activate only one policy at a time.

You can define as many service policies as needed by the installation. For example, other service policy names might be SHIFT2, SHIFT3, HOLIDAY, and WEEKEND. Each name describes what type of transactions executes during that service policy.

2.8 New service policy

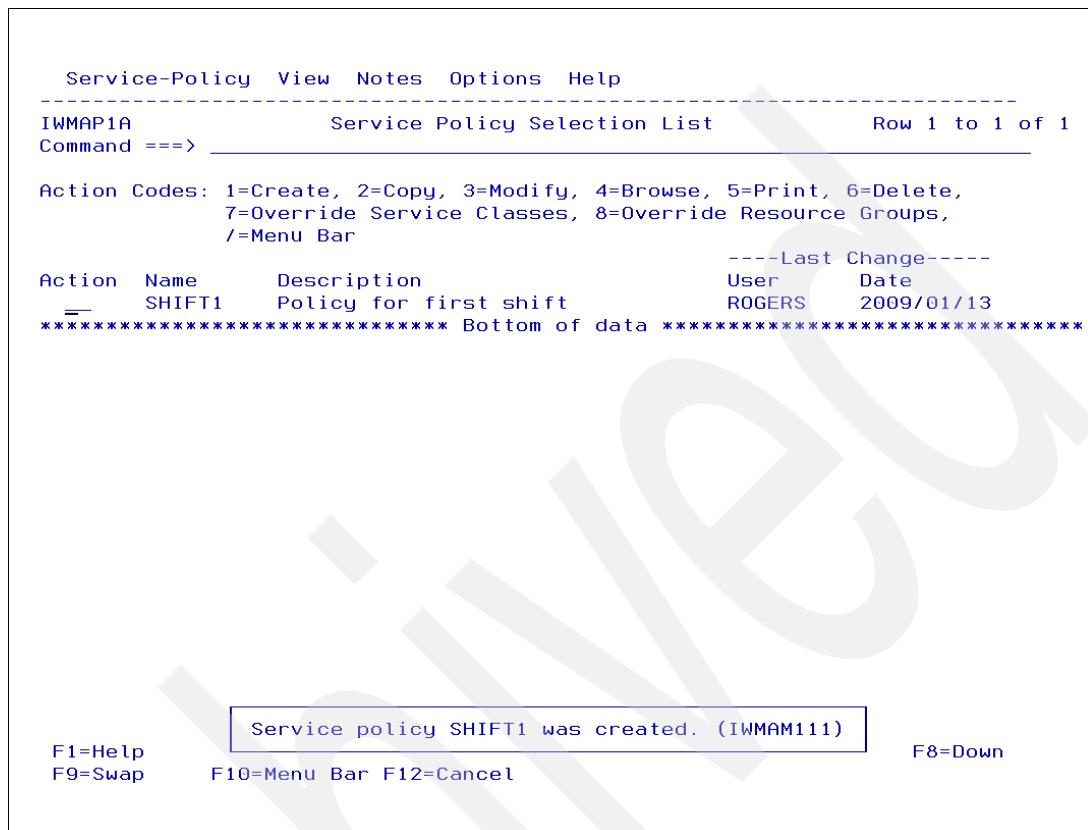


Figure 2-9 Creating a service policy

Creating the service policy

This panel (IWMAP1A), as shown in Figure 2-9, lists the defined service policies in the current service definition, which is now the new one just created. From this panel you can create, copy, modify, browse, override, or delete a service policy.

When you are creating your service definition, you may choose to define one empty “default” policy with no overrides at all. Next, create your workloads and service classes. Then determine how and when your service classes may have different goals at different times. Define additional policies with the appropriate overrides for these time periods.

Service policies

Every service policy name must be unique in a service definition. The service policy is activated by name in one of the following ways:

- ▶ By using an operator command from the z/OS operator console.
- ▶ By using a utility function from the workload management ISPF application.

To create a new service policy, select option 1 on the panel shown in Figure 2-7 on page 47. Create as many service policies as your installation needs. The service policy name in Figure 2-9 is called SHIFT1. You might then have a different service policy for each shift during the day with the same workloads and same service classes, but with different goals to execute a different type of work.

2.9 Creating a workload

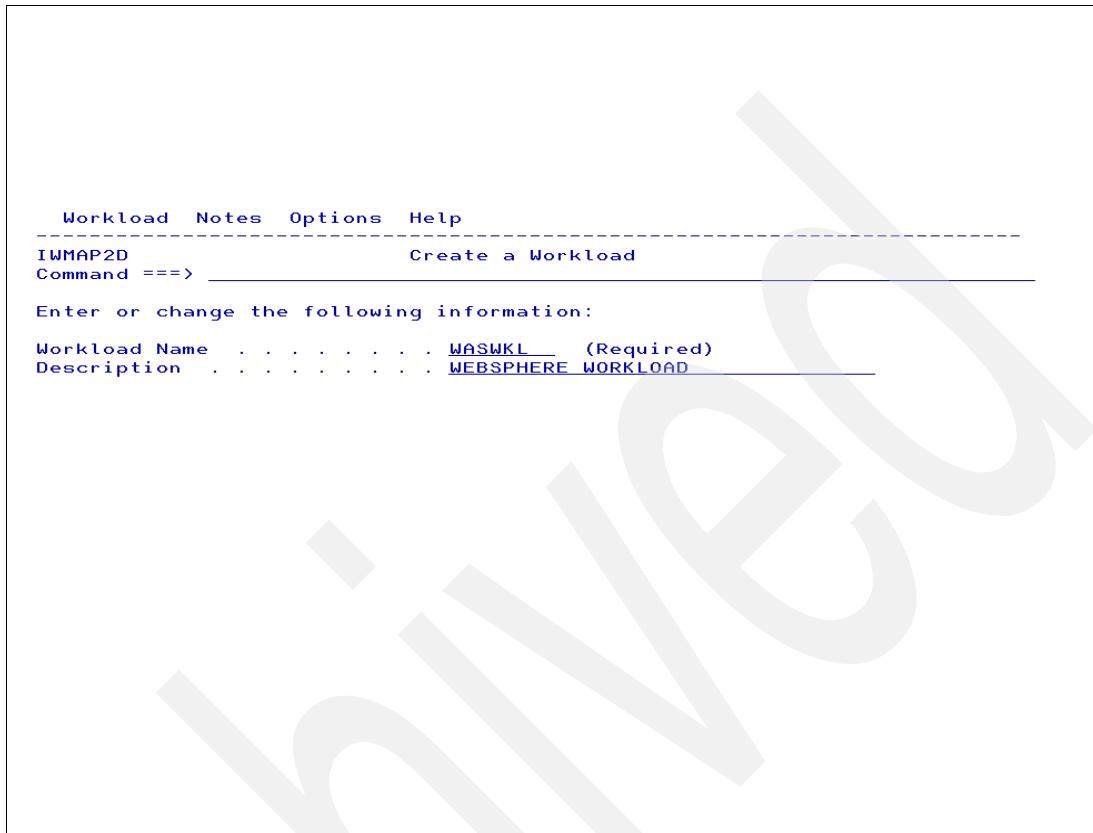


Figure 2-10 Creating a workload

Workloads

A workload is a unit of accounting comprised of related service classes (SCs). You reach this panel (IWMAP2D), as shown in Figure 2-10, by entering option 2 in the SD panel (IWMAP01).

Next, create a new workload, WASWKL, for WebSphere transactions. In the workload name field, specify a description of the workload available to performance monitors for reporting. Define as many workloads as your installation requires. Workloads contain a set of SCs for accounting purposes. You associate a workload with an SC in the SC panel.

RMF reports

RMF reports data about different application transactions grouped into categories that you have defined as workloads and SCs in your service policy. The appropriate grouping of application transactions is important. In addition to using workloads, the installation may use report classes as unit of accounting and reporting, as follows:

- ▶ Different applications which should be managed according to the same goals, you should define same SC. Applications with different goals need to be assigned to different SC.
- ▶ If you want to get separate reporting for different applications in the same SC, you can define separate report classes (RC) for each of them. Reporting for RC is possible with the same level of detail (RC period) as for SCs.

Note: You must define at least one workload before selecting the service class option.

2.10 Defining service class goals

- Choose a goal type for the performance period
 - Choose an importance number (1 to 5)
 - Choose a duration
- Choose a goal type for each period you define:
1. Average response time
 2. Response time with percentile
 3. Execution velocity
 4. Discretionary

Figure 2-11 Defining service class goals

Defining service classes

After the workloads have been created, you can begin to define service classes. To define a service class, select option 4 on the primary panel shown in Figure 2-7 on page 47. When you select option 4 from this panel, Figure 2-12 on page 54 is displayed (IWMAP3D). The figure shows the type of goals that can be defined for a performance period.

Defining goal types

There are four goal types, as explained here:

Average Response Time Average response time for transactions completing within the period in terms of hours, minutes, and seconds. Decimal points are accepted. Response time varies from 15 milliseconds to 24 hours.

Response Time and Percentile A percentile of the transactions to be completed in the specified amount of time. Percentile boundaries vary from 1 to 99. Amount of time is in hours, minutes, or seconds. Decimal points are accepted. Response time ranges from 15 milliseconds to 24 hours.

Velocity Measure of how fast transactions should run when ready, without being delayed for WLM-managed resources (as

PU, I/O, main storage and server address spaces availability). Velocity ranges from 1 to 99.

Discretionary

With discretionary goals, workload management decides how best to run this work. Because workload management's prime responsibility is matching resources to work, use discretionary goals for work for which you do not have a specific performance goal. For a service class with multiple performance periods, you can specify discretionary only as the goal in the last performance period.

Discretionary work is run using any system resources not required to meet the goals of other work. If certain types of other work are overachieving their goals, that work may be "capped" so that the resources may be diverted to run discretionary work.

Importance number

The relative importance of the service class goal is a reflection of how important it is that the service class goal be achieved. Workload management uses the importance figure only when work is not meeting its goal. Importance indicates the order in which transactions work should receive resources (priorities) when those transactions are not achieving their goals. Importance is required for all goal types except discretionary. Importance applies on a performance period level and you can change importance from period to period. Importance is stated in five levels, from 1 to 5, with 1 being the highest importance.

Duration of goal type

Duration is the amount of service that transactions running in this period should consume before going on to the next period (with a different goal). Duration is specified in service units. You do not specify a duration on the last defined period. The period concept is designed to protect trivial transactions (the 80% of all transactions that consume only 20% of the resources).

2.11 Creating a service class (1)

```
Service-Class Notes Options Help
-----
IWMAP3D           Create a Service Class      Row 1 to 1 of 1
Command ==>
-----
Service Class Name . . . . . WASPROD (Required)
Description . . . . . WebSphere SC production
Workload Name . . . . . WASWKL (name or ?)
Base Resource Group . . . . .
Cpu Critical . . . . . NO (YES or NO)

Specify BASE GOAL information. Action Codes: I=Insert new period,
E>Edit period, D>Delete period.

---Period--- -----Goal-----
Action # Duration Imp. Description
***** Bottom of data *****
```

Figure 2-12 Creating a service class

Creating a service class

After you define your workloads, you can define your service classes. You reach this panel (IWMAP3D) by entering option 4 in the Service Definition panel (IWMAP01). Create a new service class called WASPROD.

Figure 2-12 shows a Create a Service Class panel. You must assign a workload to the service class in the Workload field.

Specify the one- to eight-character name of the service class. Every service class name must be unique within a service definition. You can specify more information about the service class in the description field.

Service class fields

To create new service class fields, enter a name, a description, associated workload, and a base goal. Optionally, you can specify a resource group.

Note: When you create a new service class, it automatically belongs to all defined service policies.

2.12 Creating a service class (2)

```
Service-Class Notes Options Help
-----
IWMAP3D          Create a Service Class      Row 1 to 3 of 3
Command ==> _____
-----  
Service Class Name . . . . . WASPROD (Required)
Description . . . . . WebSphere SC production
Workload Name . . . . . WASWKL (name or ?)
Base Resource Group . . . . . (name or ?)
Cpu Critical . . . . . NO (YES or NO)  
Specify BASE GOAL information. Action Codes: I=Insert new period,
E>Edit period, D>Delete period.  
-----Period-----Goal-----  
Action # Duration Imp. Description  
____ 1 200 1 Average response time of 00:00:05.000  
____ 2 Discretionary  
***** Bottom of data *****
```

Figure 2-13 Service class period definition

Service class period goal

After the name, description, and workload name, you must define the goals for each service class period by entering option 1 on any line in the Action column. The goal selection pop-up is displayed, as shown in Figure 2-14. From this panel, select the type of goal you want to assign to the service class.

- 1. Average response time
- 2. Response time with percentile
- 3. Execution velocity
- 4. Discretionary

Figure 2-14 Select a goal type option

When you choose option 1, Average response time, the application displays the average response time goal pop-up. As shown in Figure 2-15 on page 56, there is a different pop-up for each goal type, where you can fill in the information for the goal.

Important: If you are defining a single period goal, then do not fill in a duration. If you are defining multiple periods, then you must fill in a duration.

In addition, multiple periods are not supported for work in the IMS and CICS subsystem work environments because service units are accumulated to the address space, not the individual transactions. Therefore, the system cannot track a duration for those transactions.

Average response time goal
Enter a response time of up to 24 hours for period 1
Hours <input type="text"/> (0-24)
Minutes <input type="text"/> (0-99)
Seconds <input type="text"/> 5 <input type="text"/> (0-9999)
Importance . . . 1 (1=highest, 5=lowest)
Duration . . . 200 (1-999,999,999, or none for last period)

Figure 2-15 Choosing the average response time goal (option 1)

The goal chosen in Figure 2-15 shows that transactions should complete in the first period within 5 seconds, and that WLM monitors these transactions with an importance of 1. All transactions consuming more than 200 service units are automatically migrated to the second period.

Service class second period definition

Because in the service class first period the DUR value is 200 service units, the WLM application forces you to define a new period.

In this second period the goal is Discretionary and there is no DUR, indicating that this is the last period of the WASPROD service class.

2.13 System-provided service classes

- ❑ Default service classes
 - SYSTEM
 - Address spaces include: MASTER, TRACE, GRS, DUMPSRV, SMF, CATALOG, RASP, XCFAS, SMXC, CONSOLE, IOSAS, and others
 - SYSSTC
 - For all started tasks not otherwise associated with a service class
 - SYSSTC1-SYSSTC5
 - Provided for future z/OS and EWLM support
 - SYSOTHER
 - For all other work not associated with a service class

Figure 2-16 System-provided service classes

System service classes

If some work comes into a system for which there is no associated service class defined in the classification rules, workload management assigns it to a default service class. You do not need to set up service classes for these system address spaces. Workload management recognizes these as special system address spaces and treats them accordingly. There are several such default service classes, as listed here:

SYSTEM For all system address spaces designated “high dispatching priority” (X'FF) address spaces. The high dispatching priority address spaces include: MASTER, TRACE, GRS, DUMPSRV, SMF, CATALOG, RASP, XCFAS, SMXC, CONSOLE, IOSAS, and others. For a list of the high dispatching priority address spaces in your installation, see the RMF Monitor II report and look for the x'FF' dispatching priority.

Note: To ensure that the system runs smoothly, certain address spaces cannot be freely assigned to all service classes. The following address spaces are always classified into service class SYSTEM, independent of the user-defined classification rules: *MASTER*, INIT, WLM, XCFAS, GRS, CONSOLE, IEFSCHAS, IXGLOGR, SMF, CATALOG, SMSPDSE, and SMSPDSE1.

SYSSTC For all started tasks not otherwise associated with a service class. Workload management treats work in SYSSTC just below special system address

spaces in terms of dispatching. You can also assign work to the SYSSTC service class as part of your work classification rules. You can do this for classification rules in the following subsystem types:

ASCH, JES, OMVS (z/OS UNIX System Services), STC, and TS0

Note: Address spaces in the SYSSTC service class are kept at a very high dispatching priority.

You can also assign address spaces to the SYSTEM and SYSSTC service classes as part of your work classification rules.

- | | |
|-----------------|---|
| SYSSTCx | The service classes SYSSTC1, SYSSTC2, SYSSTC3, SYSSTC4, and SYSSTC5 are provided for future z/OS and EWLM support. Service class SYSSTC and the SYSSTCx service classes are congruent as far as management of work is concerned. Work assigned to any of these service classes is managed identically to work assigned to any other. Currently, there is no technical reason to choose SYSSTCx as an alternative to SYSSTC. Many displays (for example, SDSF displays) provide the service class assigned to an address space. It is possible that you might assign SYSSTCx to convey some meaning that would be similar to a report class. |
| SYSOTHER | For all other work not associated with a service class. This is intended as a "catcher" for all work whose subsystem type has no classification. It is assigned a discretionary goal. |

SYSTEM and SYSSTC service classes

Use the system-provided service classes SYSTEM, and SYSSTC for your STC service classes. WLM recognizes certain system address spaces when they are created (such as GRS, SMF, CATALOG, MASTER, RASP, XCFAS, SMXC, CONSOLE, IOSAS, WLM), puts them into the SYSTEM service class, and treats them accordingly. If a started task is not assigned to a service class, WLM manages the started task in the SYSSTC service class. Started tasks in SYSSTC are assigned a high dispatching priority. This is appropriate for started tasks such as JES and VTAM.

Note that not all started tasks are appropriate for SYSSTC, because a CPU-intensive started task could use a large amount of processor cycles. However, if your processor is lightly loaded, SYSSTC might be appropriate because that one task may not affect the ability of the remaining processors to manage the important work with goals.

2.14 Address space dispatch priority

- ❑ Work is dispatched by priority
 - Range of dispatch priority is 191 to 255
 - Range from 202 to 207 is not used
 - Range from 192 to 201 is used for discretionary work
- ❑ Service policy can determine dispatch priority
- ❑ Other service classes
 - Dispatch priorities are dynamically assigned to user-defined service classes
 - All dispatchable units of the same service class period have the same base dispatching priority
 - Except discretionary class work

Figure 2-17 Dispatching priorities

Dispatching priority of address spaces

Dispatching of work is done on a priority basis. That is, the ready work with the highest priority is dispatched first. The total range of priorities is from 191 to 255. It indicates the priority of dispatchable units for getting CPU service. In goal mode, dispatching priorities are not assigned by the installation, but by WLM. Figure 2-17 lists the dispatching priorities.

All dispatchable units of the same service class period have the same base dispatching priority (with the exception of the mean-time-to-wait group for discretionary work). The range from 202 to 207 is not used and the range from 192 to 201 is used for discretionary work. All discretionary work is managed by a mean time to wait algorithm, which, simplified, gives a higher dispatch priority for work using I/O and a lower dispatch priority to work demanding a lot of CPU. The last used dispatch priority is 191. It is assigned to quiesced work. That also means that quiesced work can run if the CPU is not busy. It also means that all other work can access the CPU before it.

Work initiation

When work is initiated, the control program creates an address space. All successive steps in the job execute in the same address space. The address space has a dispatching priority, which is normally determined by the control program. The control program will select, and alter, the priority in order to achieve the best load balance in the system, that is, in order to make the most efficient use of processor time and other system resources.

The dispatch priority of 255 is reserved to keep the system running and to provide system work preferential access to the CPU. All system work should run in the predefined service class SYSTEM, which always gets a dispatch priority of 255. Support work and system-related work can be associated with another predefined service class SYSSTC. Work in SYSSTC gets dispatch priority 254.

Other service classes

Below SYSTEM and SYSSTC service classes starts the range of dispatch priorities that are dynamically assigned to user-defined service classes. These are all service classes with an importance of 1 to 5. One special dispatch priority is used for service classes that consume very little CPU service. Because it is difficult for WLM and not beneficial for the whole system to deal with work that only requires very little CPU, WLM gives this work a dispatch priority just below the two system-related dispatch priorities.

All dispatchable units of the same service class period have the same base dispatching priority (with the exception of the mean-time-to-wait group for discretionary work). The dispatching order of all dispatchable units with the same dispatching priority in the same or in different Service Class periods are periodically rotated.

The range from 202 to 207 is not used and the range from 192 to 201 is used for discretionary work. All discretionary work is managed by a mean time to wait algorithm, which simplified gives a higher dispatch priority for work using I/O and a lower dispatch priority to work demanding a lot of CPU. The last used dispatch priority is 191. It is assigned to quiesced work. That also means that quiesced work can run if the CPU is not busy. It also means that all other work can access the CPU before it.

Task priority

Each task in an address space has a limit priority and a dispatching priority associated with it. The control program sets these priorities when a job step is initiated. The dispatching priorities of the tasks in an address space do not affect the order in which the control program selects jobs for execution because that order is selected on the basis of address space dispatching priority.

SRM defines dispatching priority for service class periods. All address spaces in a service class period have the same base dispatching priority. Multiple service class periods may have the same base dispatching priority. After a dispatching priority change, service class periods may be remapped to different dispatching priorities such that there is an unoccupied priority between each occupied priority. This process is referred to as *priority unbunching*.

2.15 Subsystem list for classification rules

Subsystem-Type View Notes Options Help							
IWMAP7A			Subsystem Type Selection List for Rules		Row 1 to 14 of 14		
Command ==> _____							
Action Codes: 1=Create, 2=Copy, 3=Modify, 4=Browse, 5=Print, 6=Delete, /=Menu Bar							
Action	Type	Description	-----Class-----	Service	Report		
	ASCH	APPC Transaction Programs		ASCH			
3	CB	WebSphere/Component Broker		DDFDEF	RCB		
	CICS	CICS Transactions		CICS			
	DB2	DB2 Sysplex Queries		DB2QUERY			
	DDF	DDF Work Requests		DDFBAT			
	EWLM	EWLM Subsystem for ESC/ETC		SC_EWLM	REWLMDLF		
	IMS	IMS Transactions		IMS			
	IWEB	Web Work Requests			RIWEB		
	JES	Batch Jobs		BATCHLOW	BATCHDEF		
	MQ	MQSeries Workflow			RMQ		
	OMVS	UNIX System Services		SYSSTC1			
	STC	Started Tasks		STC	RSYSDFLT		
	SYSH	linux					
	TSO	TSO Commands		TSO	TSO		
***** Bottom of data *****							

Figure 2-18 Subsystem list for classification rules

Classification rules

Classification rules are used by workload management to associate a performance goal with transactions by associating incoming ones with a service class. When you set up your service definition, you identify the workloads, the resource groups, the service classes, and the service class periods containing goals based on your performance objectives. Then you define classification rules and one or more service policies. This information makes up the base service definition. Using the WLM application, you need to add classification rules to create rules associating external properties of the arrival transaction with a service class, which contains the goals used by WLM to manage the transaction priorities.

First rules panel

The set of rules are divided per transaction manager types that WLM calls “subsystem.”

The panel IWMAP7A, shown in Figure 2-18, displays the subsystem types in the current service definition. You reach this panel by entering option 6 in IWMAP01. Classification of transactions depends on having the rules defined for the correct subsystem type. This panel initially contains the reserved names of the IBM-supplied subsystem types.

Use the Modify option, option 3, to create rules for the IBM-supplied subsystem types. Option 1 is used to create a new subsystem. In Figure 2-18, a Modify on the subsystem type CB (Component Broker) subsystem is selected to create classification rules for all WebSphere transactions in your installation. Eventually, you create rules for all subsystems used by your installation.

2.16 Classification qualifiers to classify work

Work qualifiers			
➤ Each subsystem uses some of these qualifiers			
➤ Full list of work qualifiers and their abbreviations are:			
AI	Accounting information	PR	Procedure name
CI	Correlation information	PRI	Priority
CN	Collection name	PX	Sysplex name
CT	Connection type	SE	Scheduling environment name
ESC	EWLM service class name	SI	Subsystem instance
ETC	EWLM transaction class name	SPM	Subsystem parameter
LU	Logical Unit name	SSC	Subsystem collection name
NET	Netid	SY	System name
PC	Process name	TC	Transaction class/job class
PF	Perform	TN	Transaction name/job name
PK	Package name	UI	Userid
PN	Plan name		

Figure 2-19 Work qualifiers used by subsystems to classify work

Classification work qualifiers

Classification rules are the rules you define to categorize work into service classes, and optionally report classes, based on work qualifiers. A work qualifier is what identifies a work request to the system. The first qualifier is the subsystem type that receives the work request. There is one set of classification rules in the service definition for a sysplex. They are the same regardless of what service policy is in effect; a policy cannot override classification rules. Define classification rules after you have defined service classes, and ensure that every service class has a corresponding rule.

The full list of work qualifiers and their abbreviations is shown in Figure 2-19.

Work qualifiers

Not all work qualifiers are valid for every subsystem type because they are subsystem-dependent. For many of these qualifiers, you can specify classification groups by adding a G to the type abbreviation; for example, a transaction name (TN) would become a transaction name group (TNG).

Using qualifiers during classification

The name field for work qualifiers is 8 characters in length. You can use nesting for work qualifiers longer than 8 characters. These qualifiers are accounting information, correlation information, collection name, procedure name, and subsystem parameter.

You can use masking and wild card notation to group qualifiers that share a common substring. For accounting information and the subsystem parameter, you can use a start position to indicate how far to index into the character string. When no start parameter is specified, WLM matches the name field for work qualifiers according to the number of characters specified.

Rules example

In the example shown in Figure 2-20, assume you want to associate all JES2 work from department IRS with the service class JESFAST. You assigned the default for JES2 work as service class JESMED. If all JES2 accounting information from department IRS has the characters "DIRS" starting in the seventeenth position, you enter a rule with qualifier DIRS* to match on just the four characters. If you want to filter out those jobs with the eight characters "DIRS" starting in the seventeenth position, you need another rule with qualifier DIRS to assign those jobs to JESMED.

Subsystem Type : JES					
Description A11 JES2 service classes					
-----Qualifier-----			-----Class-----		
Type	Name	Start	Service	Report	
1 AI	DIRS	17	DEFAULT: JESMED	JESMED	_____
1 AI	DIRS*	17		JESFAST	_____

Figure 2-20 Use of a wildcard

2.17 Classification group qualifiers

□ Work group qualifiers	
➤	Each subsystem uses some of these qualifiers
➤	Full list of work group qualifiers and abbreviations are:
CTG	Connection type
LUG	Logical Unit name
NETG	Netid
PKG	Perform
PNG	Package name
SIG	Plan name
SYG	Subsystem instance
TCG	System name
TNG	Transaction class/job class
UIG	Transaction name/job name
UIG	Userid

Figure 2-21 Work group qualifiers used by subsystems to classify work

Work qualifiers by group

Groups are available for grouping together work qualifiers to make classification simpler. You can create groups to collect together work when you do not have a standard naming convention that allows masking or wildcarding. A group is a collection of the same work qualifiers. For example, you may want to create a group of started tasks because you want to assign them all to the same service class.

Groups are allowed for the following work qualifiers:

- ▶ Connection type
- ▶ LU name
- ▶ Netid
- ▶ Package name
- ▶ Perform
- ▶ Plan name
- ▶ Subsystem instance
- ▶ System Name
- ▶ Transaction class
- ▶ Transaction name
- ▶ Userid

Using groups

Qualifier groups of more than five members are quicker to check than single instances in the classification rules. So if you have, for example, a long list of CICS or IMS transaction names that you want to group in a service class or report class, consider setting up a group.

If you want to assign a large number of CICS transactions to the same service class, you can create a transaction name group (TNG). You name the group (for example, CICSCONV) and list all the transaction names you want included in the group; see Figure 2-22.

```
Qualifier type . . . . . : Transaction Name
  Group name . . . . . . . . . CICSCONV (required)
  Description . . . . . . . . . CICS Conversational Group
--Qualifier--
  Name
  CDBC
  CDBI
  CDBM
  CEBR
  CECI
  CECS
  CEDA
  CEDB
  Qualifier type . . . . . : Transaction Name
  Group name . . . . . . . . . CICSLONG (required)
CDBO
  CSGX
  CSNC
  CSNE
  CSSX
```

Then you use those group names in the classification rules, as shown in this panel:

Subsystem Type : CICS			Description CICS transactions	
-----Qualifier-----			-----Class-----	
Type	Name	Start	Service	Report
1	TNG	CICSCONV	_____	DEFUALTS: CICSMED
1	TNG	CICSLONG	_____	CICSCONV
				CICSLONG

Figure 2-22 Classification rules showing work groups, CICSCONV and CICSLONG

2.18 External properties for WebSphere transactions

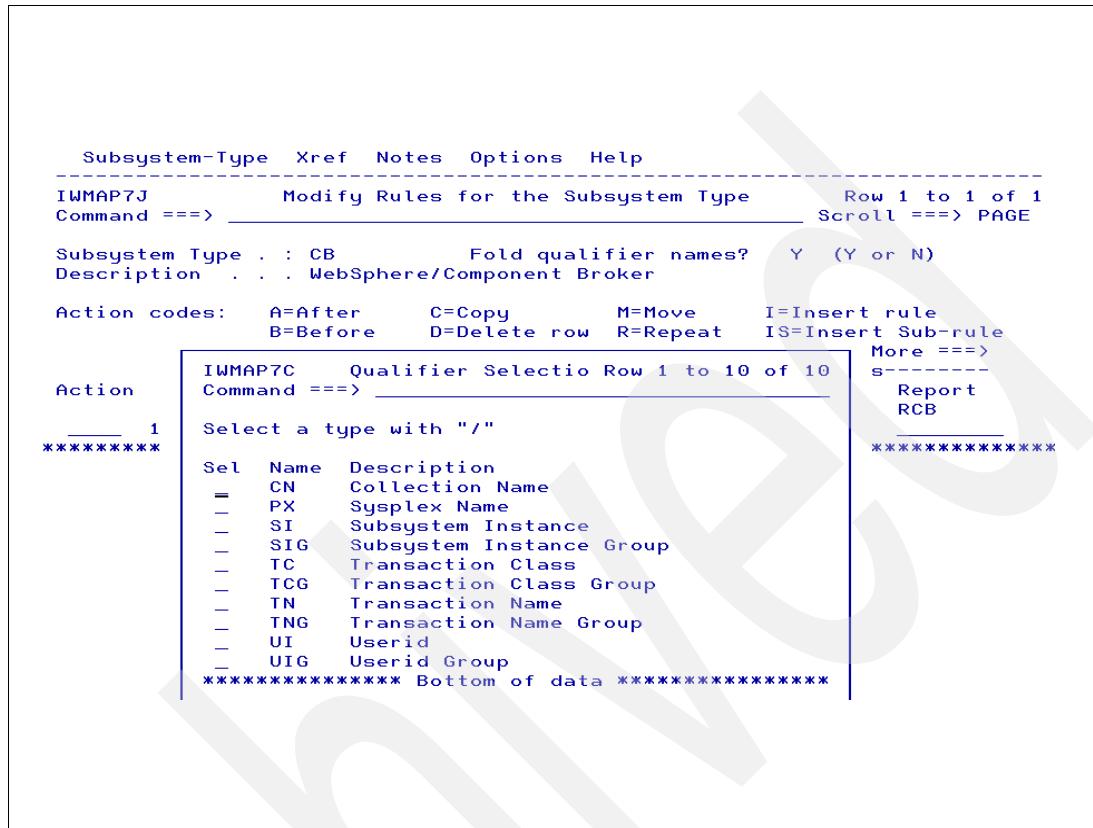


Figure 2-23 External properties for WebSphere transactions

External properties for subsystem CB (WebSphere)

Type a question mark (?) in Subsystem Type (which is covered by the pop-up window in Figure 2-23) in the classification rules for CB. The IWMAP7C panel will display, showing a list of all possible external properties variables that you are able to create a rule associating those variables with an SC. Use the F1 key to obtain detailed help information about the external properties (also called work qualifiers) grouped by subsystem type. Move the cursor to one of the following phrases, which list the subsystem type you are interested in, and press the Help key. For the CB subsystem type, the work qualifier information is listed here:

SUBSYSTEM INSTANCE CB subsystem name, or the region names

SYSPLEX NAME Sysplex name may be specified in several places, but all specifications must agree, as follows:

- The COUPLExx member of parmlib
- The LOADxx member of parmlib
- The XCF couple data set format utility

TRANSACTION CLASS A parameter on CB transactions

TRANSACTION NAME Another parameter on ICB transactions

USERID The user ID specified at LOGON time

Collection Name The set of CB objects grouped together and run in a server

2.19 Create classification rules for WebSphere

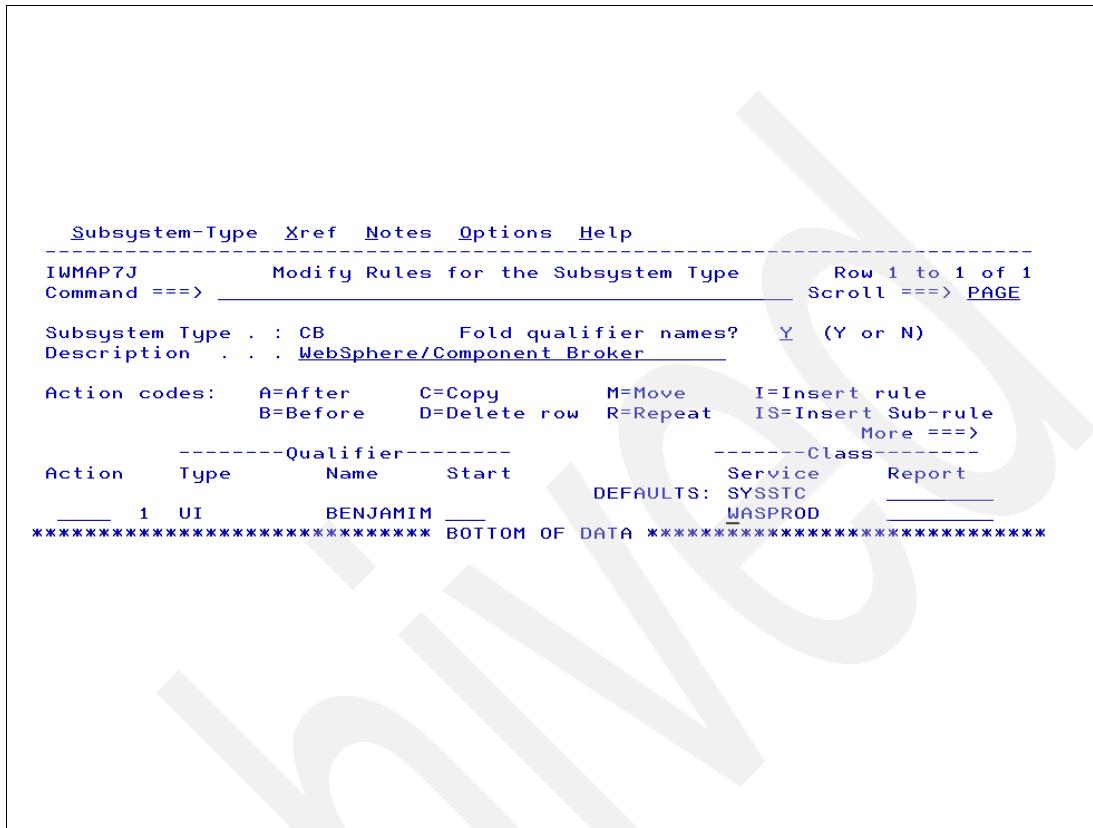


Figure 2-24 Initial panel to create classification rules for WebSphere

Classification rules panel

From the panel shown in Figure 2-24 you can create the rules associating transaction external properties variables with an specific service class. A default service class can also be defined for transactions that do not match the rules.

Transaction qualifiers

The classification rules for a subsystem are specified in terms of transaction qualifiers such as job name or transaction class. These qualifiers identify groups of transactions that should have the same performance goals and importance. The attributes of incoming transactions are compared to these qualifiers and, if there is a match, the rule is used to assign a service class to the work. A subsystem can also have a default class for work that does not match any of the rules.

This example uses the external property User Identification (UI). If the transaction is created by a user named Benjamim the service class is WASPROD. If not, the service class is the SYSSTC (automatically defined by WLM).

Note: For more detailed information about each of the work qualifier types listed, refer to "Defining Work Qualifiers" in "Defining Classification Rules" of *z/OS MVS Planning: Workload Management*, SA22-7602.

2.20 Classification rules for CICS

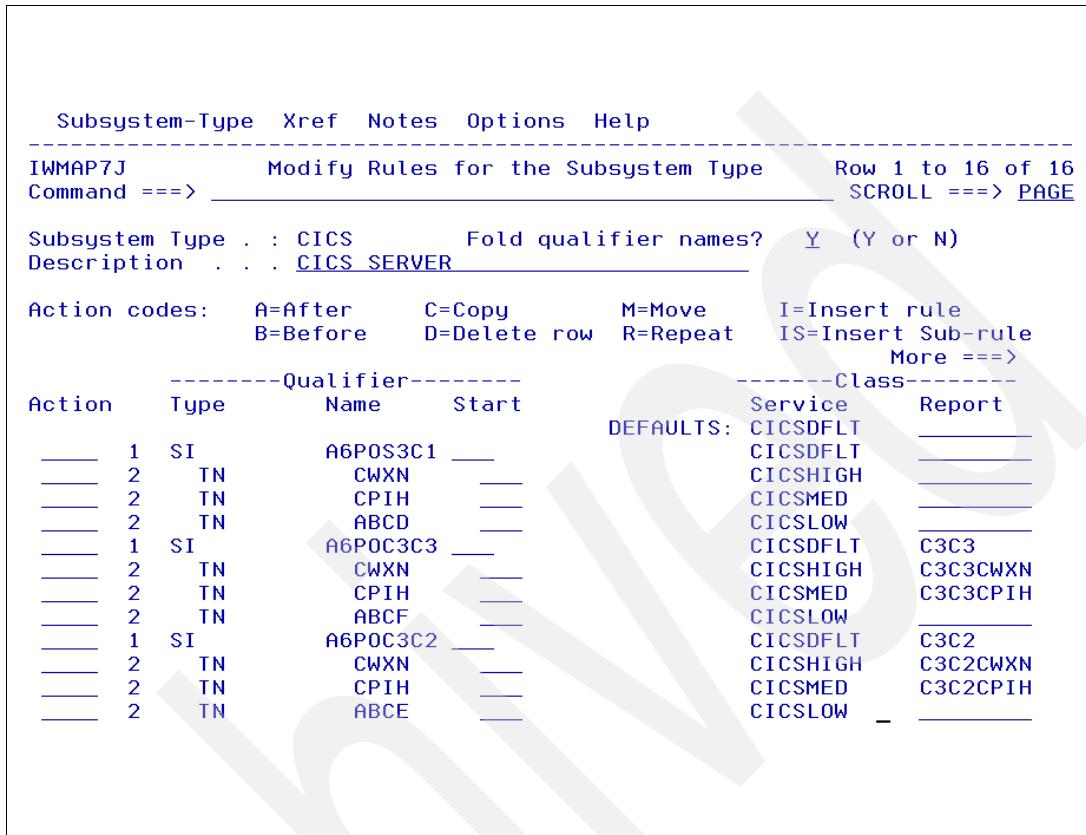


Figure 2-25 Sample classification rules for CICS

CICS classification rule example

In the example shown in Figure 2-25, the work qualifiers for classification have already been created, in this case for the CICS subsystem. (Figure 2-24 on page 67 shows a classification panel for WebSphere before any rules are created.)

When the rules for a subsystem are to be created, type a question mark (?) to view the list of qualifiers that are valid for the subsystem type you have selected. The qualifiers are shown in Figure 2-26. You can use any combination of work qualifiers to classify work into a service class.



Figure 2-26 Work qualifiers for CICS

Selecting work qualifiers

Select a work qualifier from the list shown in Figure 2-26 on page 68. The work qualifiers listed on the panel are those which are valid for the particular subsystem you have selected.

Note: You only use the work qualifiers that are needed by your installation.

Group types are specified by adding G to the type abbreviation. For example, a transaction name group is indicated as TNG. For detailed information about the work qualifiers grouped by subsystem type, move the cursor to the phrase that lists the subsystem type you are interested in and then press the Help key.

Subsystem instance

You can use subsystem instance (SI) to isolate multiple instances of a subsystem. For example, use subsystem instance if you have a CICS production system as well as a CICS test system. The VTAM applid is used to specify the subsystem instance. Individual VTAM applids can have different transaction names, as shown in Figure 2-26 on page 68.

Assigning a service class

When work is being classified, WLM is given control by the subsystem using the IWMCLSFY service and the classification rules that have been defined are checked for a match to determine the service class to be assigned. This check is made by searching through the defined classification rules until a match is found. For example, if the subsystem instance (SI) name is A6POC3C3 and the transaction name is CPIH, then the service class assigned will be CICSMED.

Service class default

If no match is found during the search of the rules, then the default service class CICSDFLT is assigned for the transaction as shown in Figure 2-26 on page 68. A service class default is the service class that is assigned if no other classification rule matches for that subsystem type. If you want to assign any work in a subsystem type to a service class, then you must assign a default service class for that subsystem.

Archived

WLM goal management

This chapter explains in more detail the mechanisms used by Workload Manager to honor the goals as entered in the Service Definition Application described in Chapter 2, “WLM ISPF application” on page 39.

Additionally, all the functions executed by Workload Manager are listed here. However, because of the introductory nature of the ABCs Redbooks, not all of the functions are explained in detail. For further information see *z/OS MVS Planning: Workload Management*, SA22-7602 and *System Programmer’s Guide to: Workload Manager*, SG24-6472.

This chapter contains the following topics:

- ▶ WLM additional functions
- ▶ How WLM measures and works toward goal achievement
 - Type of goals
 - Performance Index
 - Importance
- ▶ Logic of the policy adjustment routine
- ▶ Enclave management
- ▶ PU management
- ▶ I/O management
- ▶ Storage management
- ▶ Dynamic Parallel Access Volume
- ▶ Intelligent Resource Director

3.1 WLM additional functions

- ❑ I/O priority queueing
 - Use of dispatching priority
- ❑ Enclave creation and management
- ❑ CPU and storage protection
- ❑ Use of dynamic Parallel Access Volume (PAV)
- ❑ Intelligent Resource Director (IRD)
 - Dynamic Channel Path Management
 - Channel Subsystem Priority Queuing

Figure 3-1 Additional set of WLM functions

WLM functions

WLM has several key functions in z/OS, as listed in Figure 3-1. Following is a brief description of each one.

I/O priority queuing

I/O priority queuing is used to control non-paging DASD I/O requests that are queued because the device is busy. You can optionally have the system manage I/O priorities in the sysplex based on service class goals. The default for I/O priority management is no, which sets I/O priorities equal to dispatching priorities. If you specify yes, workload management sets I/O priorities in the sysplex based on goals.

WLM dynamically adjusts the I/O priority based on how well each service class is meeting its goals and whether the device can contribute to meeting the goal. The system does not micro-manage the I/O priorities, and changes a service class period's I/O priority infrequently.

Enclave creation and management

An *enclave* is a transaction that can span multiple dispatchable units (SRBs and tasks) in one or more address spaces and is reported on and managed as a unit. Enclaves are control blocks used to keep priorities and account information at the transaction level. It is implemented for several type of transactions such as DDF, WebSphere, and DB2 Stored procedure. WLM creates an enclave when transactions of such types arrive, and it and destroys the enclave when the transactions end.

With multisystem enclave support, enclaves can run in multiple address spaces spanning multiple systems within a Parallel Sysplex. As in a single system enclave, the work will be reported on and managed as a single unit. UNIX System Services Parallel Environment uses multisystem enclaves to run parallel jobs. With all tasks of the job running in the same enclave, WLM can manage all of the work to a single performance goal. Some work managers split large transactions across multiple systems in a Parallel Sysplex, improving the transaction's overall response time. These work managers can use multisystem enclaves to provide consistent management and reporting for these types of transactions.

CPU storage protection

Several options are available to help performance administrators protect critical work. Although applicable to several other subsystem types, CICS and IMS work will particularly benefit from the enhancements described in this section:

- ▶ Long-term storage protection
- ▶ Long-term CPU protection
- ▶ Exemption from transaction response time management

You can assign storage protection to all types of address spaces using classification rules for subsystem types ASCH, JES, OMVS, STC, and TSO. By specifying yes in the "Storage Critical" field for a classification rule, you assign storage protection to all address spaces that match that classification rule. However, before it can be storage-protected, an address space must be in a service class that meets two requirements:

- ▶ The service class must have a single period.
- ▶ The service class must have either a velocity goal, or a response time goal of more than 20 seconds.

Parallel Access Volume

WLM mainly manages dispatchable unit PU dispatching priority and I/O priority (in UCB, SAP, and DASD controller queues) to honor the transaction goal as declared by the installation in the WLM policy. If the major reason for not achieving a goal is I/O delay and simply changing the I/O priority value does not provide relief, then WLM may use the dynamic Parallel Access Volume (PAV) function. Through PAV I/O parallelism at the 3390 level, WLM tries to decrease I/O delay time. In addition to defining goals, the installation may use more drastic means to protect its most important transactions. One approach is by using CPU protection, and the other approach is by using storage protection. These are both implemented in WLM through the Critical option in the WLM policy.

Intelligent Resource Director

Intelligent Resource Director (IRD) is a set of functions executed by WLM, SAP, and LPAR code. If allowed by the installation, WLM may control the number of logical CPUs in a logical partition (LP), to guarantee a minimum number required to serve the LP workload. Also, the LP weight can be managed to give CPU to a service class running only in one LP.

- ▶ Dynamic Channel Management in IRD

Dynamic Channel Management (DCM) is a technique used to implement the floating channel algorithm. Some channels are dynamically managed by WLM (through the assignment of the Director ports) to handle a surge demand in one DASD-specific controller. DCM only applies to ESCON® channels, but there is a statement of direction to implement on FICON® channels.

- ▶ Channel subsystem (for SAP) I/O priority

This IRD function is an extension of I/O priority queuing. It allows WLM prioritization of I/O requests within SAP queues by logical partition and within a logical partition.

3.2 WLM functions (1)

- Managing goals for CICS and IMS and DB2
- Scheduling Environments
- Workload balancing
- WLM z10 EC Capacity Provisioning
- HiperDispatch mode
- Managing the dispatching of TCBs and SRBs in zIIP/zAAP PUs

Figure 3-2 WLM functions (1)

Managing CICS, IMS, and DB2

To make the most of workload management, work needs to be properly distributed so that MVS can manage the resources. It is essential that the subsystems distributing work are configured properly for workload distribution in a sysplex. You do this by using the controls provided by each subsystem. For example, in a JES2 and JES3 environment, you spread initiator address spaces across each system.

Initial cooperation between MVS and the transaction managers (CICS, IMS, DB2) allows you to define performance goals for all types of MVS-managed work. Workload management dynamically matches resources (access to the processor and storage) to work to meet the goals. CICS, however, goes further with the CICSplex Systems Manager (CICSplex® SM) to dynamically route CICS transactions to meet the performance goals.

Scheduling environments

Scheduling environments helps to ensure that units of work are sent to systems that have the appropriate resources to handle them. A *scheduling environment* is a list of resource names and their required states. Resources can represent actual physical entities (such as a data base or a peripheral device) or they can represent intangible qualities such as a certain period of time (like second shift or weekend). These resources are listed in the scheduling environment according to whether they must be set to ON or set to OFF. A unit of work can be assigned to a specific system only when all of the required states are satisfied. This function is commonly referred to as *resource affinity scheduling*.

WLM workload balancing

Subsystems have automatic and dynamic work balancing in a sysplex. For example, DB2 can spread distributed data facility (DDF) work across a sysplex automatically. DB2 can also distribute work in a sysplex through its sysplex query parallelism function. CICS, TSO, and APPC cooperate with VTAM and workload management in a sysplex to balance the placement of sessions. SOMobjects® can automatically spread its servers across a sysplex to meet performance goals and to balance the work. Workload management allocates resources to meet goals of the work that arrives. System programmers must use the existing methods of routing and scheduling work for subsystems except for those listed. For subsystems not exploiting workload balancing or routing services, if you want to balance your work across all MVS images in a sysplex, the system programmer must set the routing controls to either balance the arrival of work, or to ensure that all MVS images are equal candidates for processing work.

WLM z10 EC Capacity Provisioning

Capacity Provisioning is a WLM function only available in z10 EC machines. Performance and capacity management on System z® must ensure that work is processed according to the service level agreements that are in place. Guaranteeing service levels remains a relatively static task as long as the workloads that need to be considered are sufficiently stable. However, in many environments workloads may fluctuate considerably over time. As the total workload or the mixture of workloads varies, it may become increasingly difficult to guarantee service levels. z/OS Workload Manager (WLM) allows incoming work to be classified with a performance goal and a priority that reflects the business priority of that work. WLM will try to accommodate the goals of all the work in the system. However, even with an ideal WLM service definition, it may not be possible to achieve all specified goals when the total workload increases. In this case, trade-offs must be made. WLM decides which goals may be compromised first, based on the assigned importance level. Discretionary work will be displaced first, followed by low importance work. z/OS Capacity Provisioning helps you manage the CP, zAAP, and zIIP capacity for System z10™ servers that are running one or more z/OS systems. Based on On/Off CoD, temporary capacity may be activated and deactivated with a policy you define. z/OS Capacity Provisioning simplifies the monitoring of critical workloads, and its automation features can help to activate additional resources faster than manual operation.

HiperDispatch mode

In addition to the performance improvements available with IBM System z10 processors, z/OS workload management and dispatching are enhanced to take advantage of System z10 hardware design. A mode of dispatching called HiperDispatch now provides additional processing efficiencies. HiperDispatch mode aligns work to a smaller subset of processors to maximize the benefits of the processor cache structures, thereby reducing the amount of CPU time required to execute work. Access to processors has changed with this mode. As a result, prioritization of workloads via WLM policy definitions becomes more important.

Managing dispatch of dispatchable units in zIIP/zAAP

Without HiperDispatch, for all levels of z/OS, a TCB or SRB may be dispatched on any logical processor of the type required (standard, zAAP or zIIP). A unit of work starts on one logical processor and subsequently may be dispatched on any other logical processor. The logical processors for one LPAR image will receive an equal share for equal access to the physical processors under PR/SM™ LPAR control. For example, if the weight of a logical partition with four logical processors results in a share of two physical processors, or 200%, the LPAR hypervisor will manage each of the four logical processors with a 50% share of a physical processor. All logical processors will be used if there is work available, and they typically have similar processing utilizations.

3.3 WLM functions (2)

- ❑ CPU enqueue promotion algorithms
- ❑ WLM buffer pool management and DB2
- ❑ Performance reports (IWMRCOLL API) as requested by performance monitors
- ❑ WLM console commands
- ❑ SMF 99 records

Figure 3-3 WLM complete set of functions (2)

WLM enqueue promotion

With WLM, enqueue management has been improved with a more sophisticated enqueue promotion algorithm. An address space or enclave is promoted in terms of dispatch priority when it holds a resource that another higher priority address space or enclave wants to have. The resource manager indicates this situation to SRM through an ENQHOLD sysevent. By promoting the address space or enclave for a limited amount of time, the system hopes that the holder gives up the resource faster than it usually would with a lower DP. Also, while being promoted, the system ensures that the address space or address space associated with the enclave is not swapped out.

When a contention disappears, the resource manager notifies SRM through an ENQLSE sysevent. The enqueue promotion interval can be set by the installation through the ERV option in IEAOPTxx parmlib member. The ERV option specifies the CPU service units that an address space or enclave can absorb while it is promoted before the promotion is taken back.

In goal mode, the enqueue promotion dispatch priority is determined dynamically at every policy adjustment interval (10 seconds). It can change based on available processing capacity and amount of high dispatch work in the system. Address spaces are promoted to that priority if their current dispatch priority is not already higher than that.

Buffer pool management

DB2 and WLM can fine-tune the buffer pool size based on long-term trends and steady state growth. The **DISPLAY BUFFERPOOL** command output includes an **AUTOSIZE** attribute that allows

you to enable or disable automatic buffer pool management at the individual buffer pool level. Automatic buffer pool management is off by default.

When you enable automatic buffer pool management, DB2 reports the buffer pool size and hit ratio for random reads to the WLM component. DB2 also automatically increases or decreases buffer pool size, as appropriate, by up to 25% of the originally allocated size. DB2 limits the total amount of storage that is allocated for virtual buffer pools to approximately twice the amount of real storage. However, to avoid paging, set the total buffer pool size to less than the real storage that is available to DB2.

WLM performance reports (IWMRCOLL API)

WLM has several APIs to be informed about events happening in the sysplex. There is also an interface used by performance monitors such as RMF and Omegamon to obtain WLM performance data to be used in reports.

WLM console commands

In addition to the WLM policy stored in the WLM couple data set, the installation can communicate with WLM through a set of MVS console commands such as:

Display, Vary, Modify

SMF 99 records production

WLM may produce a large amount of log information regarding the decisions it executed and the decisions that it considered but did not execute. You may download, from the Internet, free code that will enable you to reduce such records in readable reports. Consider this course of action mainly if you disagree or do not understand some WLM decisions. You may want to switch off such SMF records during normal times, because the creation of such information consumes a significant amount of CPU cycles and I/O resource.

Note: In particular, turn off SMF type 99 records. They trace the actions SRM takes while in goal mode, and are written frequently. SMF type 99 records are for detailed audit information only. Before switching your systems into goal mode, make sure you do not write SMF type 99 records unless you specifically want them.

3.4 WLM performance behavior

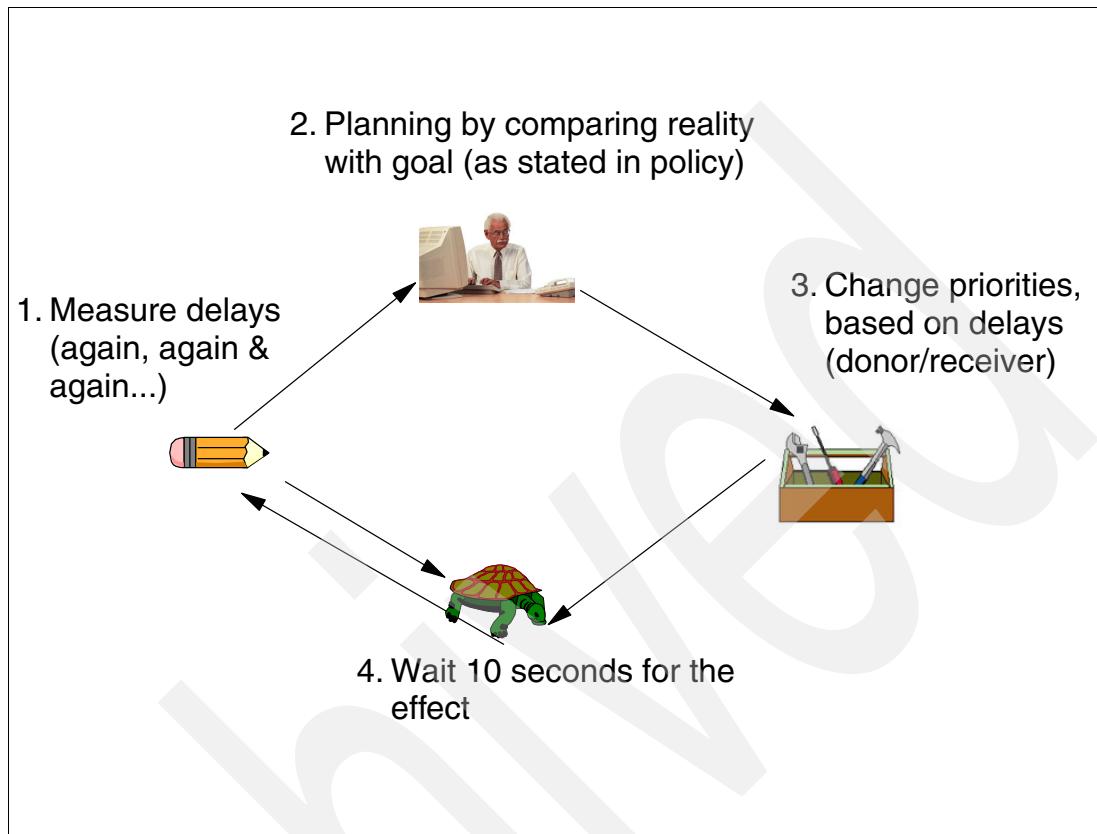


Figure 3-4 Picture of a heuristic process

Step 1. Measure delays

WLM constantly measures the systems and at certain intervals, it combines the measurement results to a complete picture, calculates how work is progressing, and compares the actual achieved results with the goal definitions to determine whether adjustments are necessary.

The system state is measured every 250 milliseconds. At these points, WLM captures the delay and using states of all work units in the system and resources it manages. WLM basically manages CPU, DASD I/O devices, system storage, server address spaces, and to some extent, resource serialization like GRS enqueues. The states are manifold. While for CPU there is currently one using and one delay state, it is possible to observe multiple possibilities why work cannot use the storage (for example, paging delays, shortages of the page data sets, and much more).

Every 10 seconds WLM summarizes system state information and combines it with measured data like CPU service or transaction end times. At this point WLM learns the behavior of the workloads and the system. It retains a certain amount of historical data in storage and maintains graphs showing the relationship between dependent functions in the system.

Step 2. Policy-defined goals

The work classification, along with the WLM observation of how the work uses resources, provides the base for managing the computer. Management is performed based on goals that

the installation defines for the work. After classifying the work into distinct classes, the installation associates a goal with each class. The goal determines how much service the work in the class is able to receive. These classes of work are known as *service classes*.

In addition to the goal, it is also necessary to define which work is more important than other work. If too much work wants to use the same resource of an operating system, WLM must know to which one it must give preference. This is done by defining an importance for each service class. The importance is a number ranging from 1 to 5. The most important work in the system is assigned a 1. Work which can suffer first when resources become constrained is assigned a 5.

You might have work running in the system for which no concrete goal must be achieved. Such work should *only* run when plenty of resources are available. Such work is classified to service classes with a discretionary or no goal, and thus tells the system that it should only run when service classes with an importance and goal do not need the resources to achieve their goals.

To manage work, the service classes must be associated with an importance and a goal, as follows:

- ▶ The importance tells WLM which service classes are preferred against others when the goals cannot be achieved.
- ▶ The goals tell WLM what must be achieved to meet user requirements.
- ▶ The following goals are possible for service classes:
 - An average response time
 - A percentile response time, meaning a certain percentage of work requests which have to end within a certain time limit
 - An execution velocity (the acceptable amount of delay that work should encounter when it runs in the system)
 - Discretionary, if no specific goal definition and no specific importance for the work in the service class exist
- ▶ The goals must be oriented toward user expectations as well as toward what is possible in a system, because WLM cannot exceed the physical limitations of the system.

Step 3. Donor and receiver

A *donor* is a service class period that donates resources to the receiver. Multiple donors may donate multiple resources to a single receiver during one policy interval. A *receiver* is the service class period that SRM is considering helping. SRM helps only one receiver during each policy interval, although it may assess multiple receivers before finding one to help.

The resources to help the receiver may also come out of what is referred to as “discretionary resources” which are those that can be reallocated with little or no effect on the system’s ability to meet performance goals.

Service class periods other than receivers and donors can also be affected by changes. These service class periods are referred to as secondary receivers and donors. SRM may decide not to help a receiver due to minimal net value for either a primary or secondary donor. If a service class period is being served by one or more address spaces, it is called the goal receiver or donor. It is the service class period with the response time goals.

To help such service class periods, SRM must donate resources to the server address spaces. The service class period that is serving a service class is called a resource receiver or donor. SRM adjusts resources for the resource donor or receiver to affect the performance of the goal donor or receiver.

Step 4. Policy adjustment

The purpose of policy adjustment is to meet service class and resource group goals. Policy adjustment is done approximately every 10 seconds. Using policy adjustment, SRM performs the following actions:

- ▶ Selects work to help
- ▶ Selects which performance bottlenecks to address
- ▶ Selects which work should donate resource to help other work
- ▶ Assesses whether it is worth it to help work

Archived

3.5 Sampling transaction WLM states

Transaction WLM states:

- Using
- Delayed
- Idle
- Unknown
- Quiesced

Figure 3-5 Transaction sampling WLM states

Transaction sampling WLM states

To enforce goals and to track the performance of all transactions in a sysplex, WLM samples the states of dispatchable units (TCB and SRB) associated with each running transaction every 250 milliseconds. Figure 3-5 lists the transaction sampling WLM states.

WLM transaction states

The subsystem work manager uses the execution delay monitoring services to tell workload management about their view of the current state of a work request, such as ready state, idle state, or waiting state. The actual state may be different. For example, a work request may be active from the subsystem's view, but might be delayed by a page fault, or for CPU access. The information is kept in performance blocks, also called monitoring environments.

The delay information shows the different states that server address spaces experience when processing transactions. The information is provided on a service and report class basis in the service or report class representing the transactions. This way, the delay states show for the transactions being processed, not for the address spaces serving the transactions.

The states include how many times the service or report class was seen active, ready, and waiting. There are several waiting states. Each of these is reported separately. Other states include whether the transactions are continued somewhere else in the system, in the sysplex, or in the network.

WLM sampling

Within each sample, the dispatchable unit associated to the transaction can be in one of the following states:

using	Using CPU (zAAP or zIIP) or DASD I/O.
delayed	Delayed (in a queue) by CPU (zAAP or zIIP), storage, I/O or a server address space service.
idle	Without transaction. It means a TSO or UNIX System Services address space without transactions, an initiator without a job, APPC wait, or all address spaces tasks in wait and at least one task with an outstanding Stimer function (to be awaked by a time event).
unknown	Delayed by a non-tracked WLM resource (as ENQ or operator) or being idle for a reason other than those listed under idle state.
quiesced	The transaction is quiesced by the RESET operator command.

In the RMF Workload Activity Report these states are shown as a percentage, and they should add up to 100%. However, in some cases, there might be an overlay between Using and Delay states, thus causing the sum to be greater than 100%.

Using and delay states

The using and delay states are measured as follows:

- ▶ CPU

WLM adds to the address space or enclave *using CPU* count (that is, the number of dispatchable units (multiple state) actively executing on a CPU, at every sample).

WLM adds, to the address space or enclave *delay CPU* count, the number of dispatchable units (multiple state) ready but being delayed by CPU, at each sample.

There are also counters for using and delay for zAAP and zIIP.

- ▶ Storage (paging and swapping)

WLM adds, to the address space *delay storage count*, the number of dispatchable units (multiple state) that have a storage delay (such as a page fault).

However, if the address space is swapped out in the OUTR queue by an WLM decision, or is in the process of being paged in due to a swap in, the delay storage counter is increased by one (single state). This is done at every sample.

- ▶ DASD I/O

Measuring I/O *using* and I/O *delay* is optional, but recommended.

To the address space or enclave *using I/O* count, WLM adds the number of dispatchable units that have an I/O operation in the connect state (that is, moving data through the channel). This is done at every sample.

To the address space or enclave *delay I/O* count, WLM adds the number of dispatchable units (multiple state) that have an I/O operation in the IOS queue or in pending state (delayed by channel, control unit, or shared DASD device) in the channel subsystem (that is, the channel program being delayed). This is done at every sample.

3.6 Performance Index

- ❑ A metric indicating how well service class periods are meeting their goals independently of the goal type:
 - PI=1 Service class period is exactly meeting the goal
 - PI>1 Service class period is missing the goal
 - PI<1 Service class period is overachieving the goal
 - Discretionary goal defined to have a fixed PI of 0.81

Figure 3-6 Performance Index

Performance index

Performance index (PI) is a calculation of how well work is meeting its defined goal. For work with response time goals, the performance index is the actual divided by goal. For work with velocity goals, the performance index is goal divided by actual. WLM maintains a PI for each service class period, to measure how actual performance varies from the goal.

A performance index of 1.0 indicates the service class period is exactly meeting its goal. A performance index greater than 1 indicates the service class period is missing its goal. A performance index less than 1.0 indicates the service class period is beating its goal. Work with a discretionary goal is defined to have a performance index of 0.81.

Because there are different types of goals, WLM needs to compare how well or how poorly transactions in one service class period are performing compared to other transactions. The comparison is possible through the use of the PI. The PI is a sort of normalized metric allowing WLM to compare transactions having a velocity goal with transactions having an average response time goal and with transactions having a percentile response time goal to determine which is farthest from the goal.

PI values

The following rules explain the meaning of the PI values:

- PI = 1 means that transactions in the SC period are exactly meeting their goal.
- PI > 1 means that transactions in the SC period are missing their goal.

- PI < 1 means that transactions in the SC period are beating their goal.

In the sysplex, every service class period may have two types of PIs:

- One or more *local PIs* - there is one local PI for every z/OS system where the SC period transaction is running, and it represents its performance in each local system in the sysplex.
- One *sysplex (or global) PI* - it represents the SC period global weighted performance across all the z/OS systems in the sysplex.

Archived

3.7 Sysplex PI and local PI

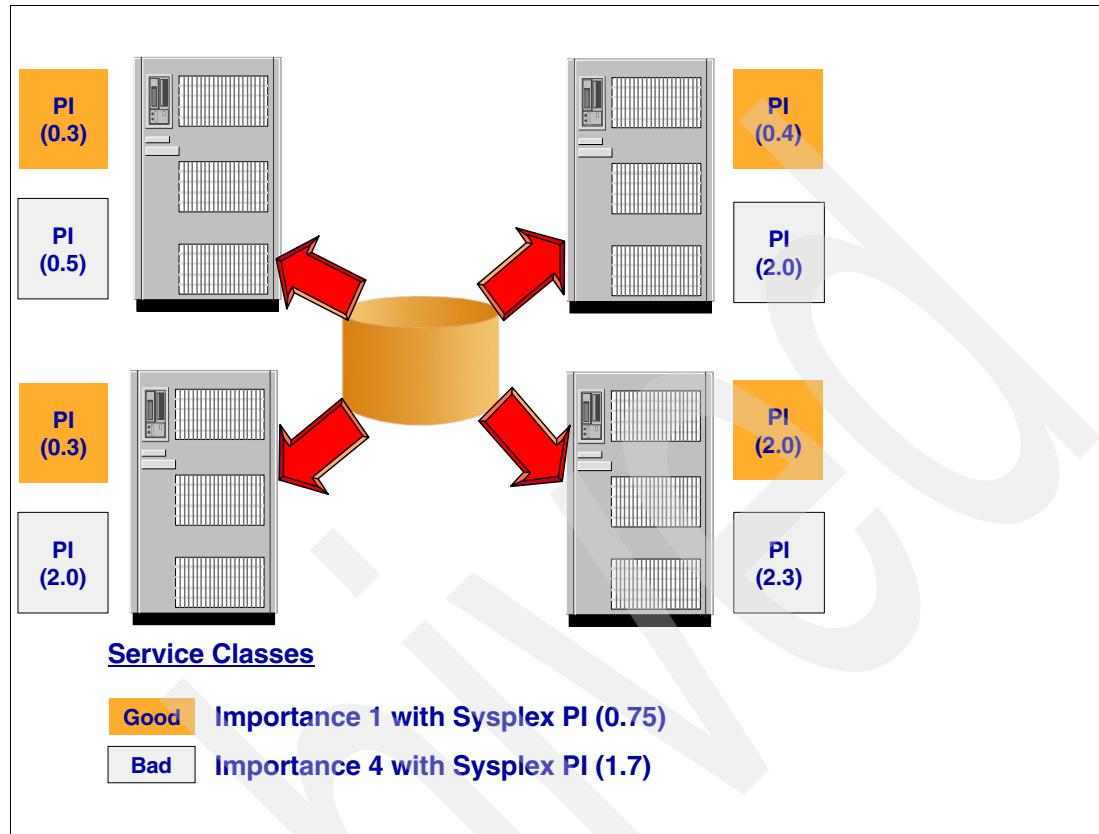


Figure 3-7 Sysplex PI and local PI

Global and local PIs

The global PI is reported in the RMF Workload Activity report and can be easily interpreted at the sysplex level. If it is below 1, it means that the goal is overachieved. If it is equal to 1, it means that the goal is exactly met. If it exceeds 1, it means that the goal is missed. Depending on the RMF options, the local PIs can also be reported.

Through services provided by XCF, WLM performance data is periodically exchanged between WLMs in a sysplex. The information is sent between the z/OS systems and allows each WLM to construct an approximate view of the status of the sysplex through the calculation of SC periods sysplex PIs. The aggregate view enables WLM algorithms such as policy adjustment to make trade-offs within each individual system. The data exchanged includes:

- ▶ Local PIs, transaction rate for each service class period, to allow each local WLM to calculate the weight average (transaction rate is the weight) sysplex PIs for each service class period.
- ▶ CPU service consumption information for each resource group, to globally enforce the resource group capping and the resource group protection algorithms. Refer to “Resource group capping” on page 154.
- ▶ Information identifying the active service policy.

Sysplex PI and local PI

The sysplex PI (also called the global PI) indicates how well a transaction is performing in a sysplex scope. This information is displayed by performance monitors (such as RMF) and it is used by each WLM for adjusting local resources.

The local PI indicates how well a transaction is performing on the local z/OS system. The local PIs are used by each WLM in a sysplex to calculate the sysplex PI at every 10 seconds.

Both PIs are significant, but it is more important to check the sysplex PI to determine whether or not the goal is achieved. The sysplex PI is a weighted average of the local PIs, where the weights are the values of transaction rates in each service class period in each z/OS system.

Performance index evaluation in a sysplex

The local PI and sysplex PI are maintained every 10 seconds for each service class period.

The following sections explain how the PI is calculated according to the type of goal. For every type of goal, the achieved value is then compared to the goal value. The achieved value is calculated, and the goal value is set by the user in the WLM ISPF application.

Figure 3-8 on page 87 shows four z/OS systems with two active service class periods. The first service class period has the following local PIs: 0.3, 0.3, 0.4, and 2.0. Its global PI is 0.75 (globally happy).

The second service class period has the following local PIs: 0.5, 2.0, 2.0, and 2.3. Its global PI is 1.7 (globally unhappy).

3.8 PI calculation - average response time goal

TRANSACTIONS		TRANS.-TIME	HHH.MM.SS.TTT	--DASD I/O--		---SERVICE---		--SERVICE RATES--		PAGE-IN RATES		---STORAGE---		
AVG	0.00	ACTUAL		157	SSCHRT	0.0	IOC	0	ABSRPTN	0	SINGLE	0.0	Avg	0.00
MPL	0.00	EXECUTION		0	RESP	0.0	CPU	0	TRX SERV	0	BLOCK	0.0	TOTAL	0.00
ENDED	432072	QUEUED		0	CONN	0.0	MSO	0	TCB	0.0	SHARED	0.0	CENTRAL	0.00
END/S	720.32	R/S AFFINITY		0	DISC	0.0	SRB	0	SRB	0.0	HSP	0.0	EXPAND	0.00
ESWAPS	0	INELIGIBLE		0	Q+PEND	0.0	TOT	0	RCT	0.0	HSP MISS	0.0		
EXCTD	0	CONVERSION		0	IOSQ	0.0	SEC	0	IIT	0.0	EXP SNGL	0.0	SHARED	0.00
AVG ENC	0.00	STP DEV		6.153					HST	0.0	EXP BLK	0.0		
REM ENC	0.00								APPL %	0.0	EXP SHR	0.0		
MS ENC	0.00													

---RESPONSE TIME---			EX	PERF	AVG	--USING--			EXECUTION DELAYS %			---DLY%--			-CRYPTO%-		
HH.MM.SS.TTT			VEL	INDX	ADRSP	CPU	I/O	TOTAL									
GOAL	00.00.00.180	ACTUALS															
*ALL	00.00.00.157		N/A	0.9	0.0	0.0	0.0	0.0				0.0	0.0	0.0	0.0	0.0	
OS0B	00.00.00.125		N/A	0.7	0.0	0.0	0.0	0.0				0.0	0.0	0.0	0.0	0.0	
OS0C	00.00.00.214		N/A	1.2	0.0	0.0	0.0	0.0				0.0	0.0	0.0	0.0	0.0	
OS0D	00.00.00.150		N/A	0.8	0.0	0.0	0.0	0.0				0.0	0.0	0.0	0.0	0.0	
OS0E	00.00.00.163		N/A	0.9	0.0	0.0	0.0	0.0				0.0	0.0	0.0	0.0	0.0	

Figure 3-8 Average response time RMF report

Average response type goal PI calculation

The average response time is calculated by dividing the sum of ended transaction response time by the number of ended transactions. This gives the current average response time field, which corresponds to the achieved performance of the service class. Then this value is divided by the goal response time to give the performance index of this service class period.

In Figure 3-9 on page 88, in the RMF Workload Activity Report, the ACTUAL field (under TRANS.-TIME) shows that the average response time is 0.157s for the 432,072 ended transactions.

The goal of the service class is 0.180, as shown in the GOAL field. The PI is calculated for the sysplex (*ALL) and for each system in the sysplex by dividing the achieved (current) average response time (either in the sysplex or on the single system) with the target average response time as illustrated in the following formula:

$$\text{PI} = \frac{\text{Average response time}}{\text{Goal average}}$$

Goal average

That corresponds for 0.9 for the sysplex PI (*ALL line) and 0.7, 1.2, and 0.8 for each local PI in the four z/OS systems (OS0A, OS 0B, OS0C, and OS0C) where those transactions are running.

3.9 PI calculation - execution velocity goal

TRANSACTIONS		TRANS.-TIME		HHH.MM.SS.TTT		--DASD I/O--		---SERVICE---		--SERVICE RATES--		PAGE-IN RATES		----STORAGE----	
AVG	2.97	ACTUAL		1.45.029	SSCHRT	899.2	IOC	223323	ABSRPTN	58154	SINGLE	0.0	AVG	3966.86	
MPL	2.97	EXECUTION		1.44.561	RESP	0.9	CPU	8820K	TRX SERV	58154	BLOCK	0.0	TOTAL	11780.8	
ENDED	12	QUEUED		468	CONN	0.6	MSO	77155K	TCB	425.0	SHARED	0.0	CENTRAL	11780.8	
END/S	0.02	R/S AFFINITY		0	DISC	0.0	SRB	153458	SRB	7.4	HSP	0.0	EXPAND	0.00	
#SWAPS	0	INELIGIBLE		0	Q+PEND	0.3	TOT	86352K	RCT	0.0	HSP MISS	0.0			
EXCTD	0	CONVERSION		262	IOSQ	0.0	/SEC	172707	IIT	3.1	EXP SNGL	0.0	SHARED	0.00	
Avg ENC	0.00	STD DEV		50.588					HST	0.0	EXP BLK	0.0			
REM ENC	0.00								APPL %	87.1	EXP SHR	0.0			
MS ENC	0.00														
VELOCITY MIGRATION: I/O MGMT 51.1% INIT MGMT 51.0%															
		---RESPONSE TIME---		EX	PERF	AVG	--USING%	----- EXECUTION DELAYS % -----		--DLY%--	-CRYPTO%-	--CNT%--		%	
		HH.MM.SS.TTT		VEL	INDX	ADRSP	CPU	I/O	TOT CAPP	CPU	I/O	UNKN	IDLE	USG	DLY
GOAL				30.0%											
ACTUALS															
ITS1					51.1%	0.6	1.0	20.1	21.7	40.0	18.0	15.4	6.6		
														18.1	0.0
														0.0	0.0
														0.0	0.0

Figure 3-9 Execution velocity goal RMF report

Execution velocity goal - PI calculation

The execution velocity percentage goal is calculated by dividing the number of the using samples by the sum of the using and delay samples.

These using and delay samples are found in the RMF Workload Activity Report. In Figure 3-9, in the RMF Workload Activity Report, the using samples percentages are found in the fields USING% CPU and USING % I/O. Their sum is 20.1% + 21.7%, giving 41.8%. The total of all the delay samples is given in the field EXECUTION DELAYS% TOT; the value is 40.0%.

The achieved execution velocity is then $41.8\% / (41.8\% + 40.0\%) = 51.1\%$.

The goal is an execution velocity of 30%. Then those transactions running in this service class period are happy. The PI is then calculated by dividing the goal execution velocity by the achieved execution velocity. In our example the calculation is $30\% / 51.1\%$, giving a PI of 0.6 (the result of the division is rounded). This formula is as follows:

$$\text{PI} = \frac{\text{Goal execution velocity}}{\text{Actual execution velocity}}$$

Notice that the formula is the inverse of the average response time PI calculation. The reason is because the higher the execution velocity figure, the better the situation is. Conversely, higher the average response time, the worse it is. In the report there is only one z/OS system, so the local and global PIs are identical.

3.10 Percentile response time - PI calculation

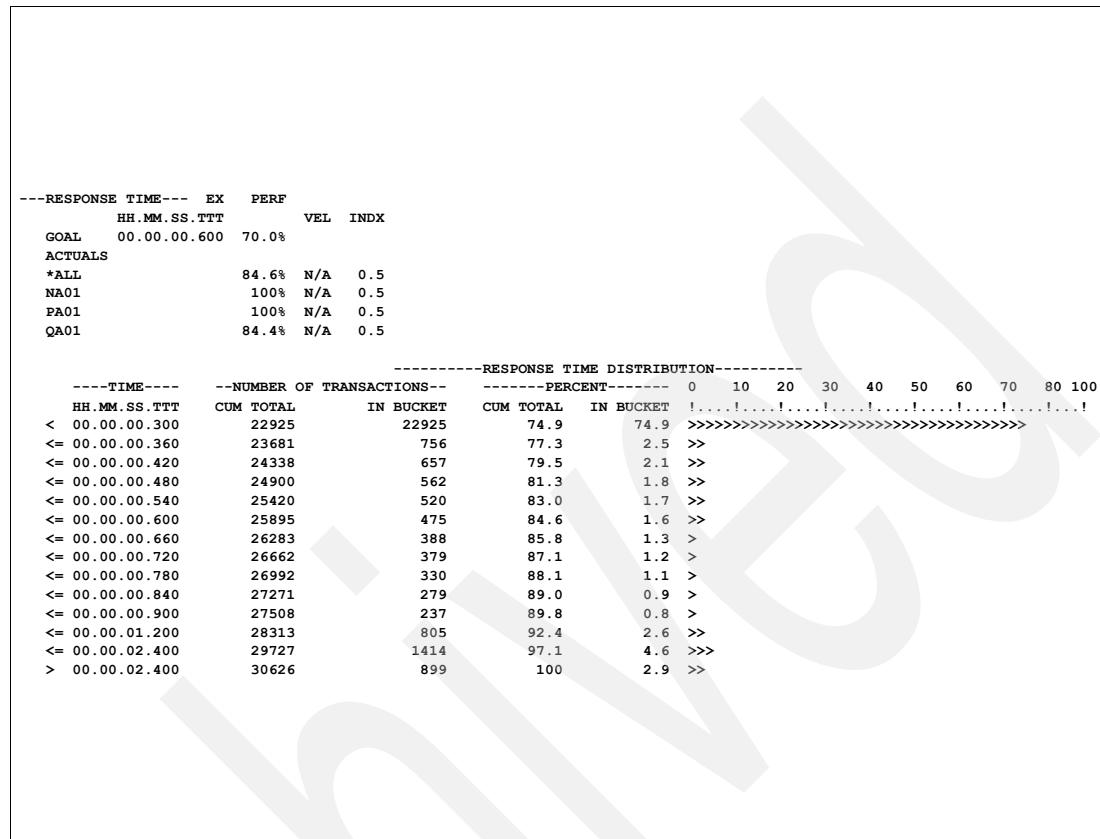


Figure 3-10 Average response time with percentile goal RMF report

Average response time with percentile goal type PI calculation

The calculation of the PI is somewhat complicated for this type of goal (RMF does it for you). In this section we explain the calculation, using the data displayed in Figure 3-10 as a base reference.

Response time distributions are provided for both service classes and report classes. These distributions consist of 14 buckets of information. There is a header, explaining the contents of the buckets, which is provided once. The header contains the value of the particular bucket, which is a percentage of the specified goal (that is, 50 equates to 50% of the goal; 150 equates to 150% of the goal). There is always one bucket that exactly maps to the specified goal, with a value of 100%. In each bucket is the number of transactions that completed in the amount of time represented by that bucket.

WLM keeps response time distribution data in buckets for service class periods that have a response time goal specified. These buckets are counters that keep track of the number of transactions ended within a certain response time range. The number of ended transactions are stored in a set of 14 buckets. These buckets exist per service class period.

- ▶ Bucket range values

Bucket values range from half of the goal response time value to more than four times the goal response time value. In our case, the goal is 70% of transactions with a response time of 0.600 seconds. The bucket values will range from 0.300 seconds to more than 2,400 seconds.

The sixth bucket contains the number of transactions that ended within the goal response time value (0.600 seconds, in our case). In our example, 84.6% of the cumulated transaction numbers ended within the goal response time value. The bucket itself shows that 475 transactions representing 1.6% of the total number of transactions have a response time between 0.540 and 0.600 seconds.

The thirteenth bucket contains the number of transactions that ended *within* four times of the goal response time value (2.400s).

The last bucket contains the transactions that ended at *more* than four times the goal response time value (more than 2.400s).

- ▶ The NUMBER OF TRANSACTION CUM TOTAL field cumulates the transactions at each bucket.
- ▶ The RESPONSE TIME DISTRIBUTION CUM TOTAL field contains the cumulated percentage of transactions at each bucket. To calculate the PI, WLM checks this fields until it finds the goal percentile, or the value immediately above that.

For example, in our case the goal percentile is 70%. The percentile immediately equal to or above is 74.9%, and it is located in the first bucket. Then WLM picks the response time of this bucket (0.300 seconds, in our case) and divides it by the goal response time, which is 0.600 seconds. The PI of this service class is then 0.5, as shown in the RMF report.

- ▶ The maximum value for a PI in a percentile goal is 4.0 because of the way the buckets are organized. Bucket 14 always corresponds to 400% of the value of the goal. The minimum value for the PI is 0.5 due to the bucket range organization.
- ▶ The general formula is then:

$$\text{PI} = \frac{\text{Percentile actual}}{\text{Percentile goal}}$$

3.11 PI calculation - discretionary goals

- The fixed performance index is equal to 0.81 and is always in the lowest dispatch priority, from 192 to 201
- Due to timely internal capping of some LPARs (PI<1), a dispatchable unit with a discretionary goal may gain CPU before LPARs with PI<1, and have a fixed I/O priority of 248
- It is the goal of the special SYSOTHER service class
- It's a universal donor and can only be defined in the last period
- It is a candidate for having its storage managed by the working set manager
- It may cause an internal CPU capping in service classes with numeric goals

Figure 3-11 Discretionary goal PI and other properties

Discretionary goals

With discretionary goals, workload management decides how best to run this work. Because workload management's prime responsibility is matching resources to work, discretionary goals are used best for the work for which you do not have a specific performance goal. For a service class with multiple performance periods, you can specify discretionary only as the goal in the last performance period.

Discretionary work is run using any system resources not required to meet the goal of other work. If certain types of other work are overachieving their goals, that work may be "capped" so that the resources may be diverted to run discretionary work.

Discretionary goal performance index

Performance index (PI) is a calculation of how well work is meeting its defined goal. For work with response time goals, the performance index is the actual divided by goal. For work with velocity goals, the performance index is the goal divided by the actual.

A performance index of 1.0 indicates the service class period is exactly meeting its goal. A performance index greater than 1 indicates the service class period is missing its goal. A performance index less than 1.0 indicates the service class period is beating its goal. Work with a discretionary goal is defined to have 0.81 as a performance index.

I/O priority

SRM defines an I/O priority for each service class period. All address spaces in a service class period have the same I/O priority. Multiple service class periods may have the same I/O priority. I/O priority is used to prioritize requests on IOS's UCB queues.

SYSOTHER service class

The SYSOTHER service class is used for all other work not associated with a service class. This is intended as a “catcher” for all work whose subsystem type has no classification. It is assigned a discretionary goal.

Universal donors

A donor is a service class period that donates resources to the receiver. Multiple donors may donate multiple resources to a single receiver during one policy interval. The resources to help the receiver may also come out of what is referred to as discretionary resources. Discretionary resources are those that can be reallocated with little or no effect on the system's ability to meet performance goals.

CPU capping

Certain types of work, when overachieving their goals, potentially will have their resources “capped” to give discretionary work a better chance to run. Specifically, work that is not part of a resource group and has *one* of the following two types of goals will be eligible for this resource donation:

- ▶ A velocity goal of 30 or less
- ▶ A response time goal of more than one minute

3.12 WLM policy adjustment routine

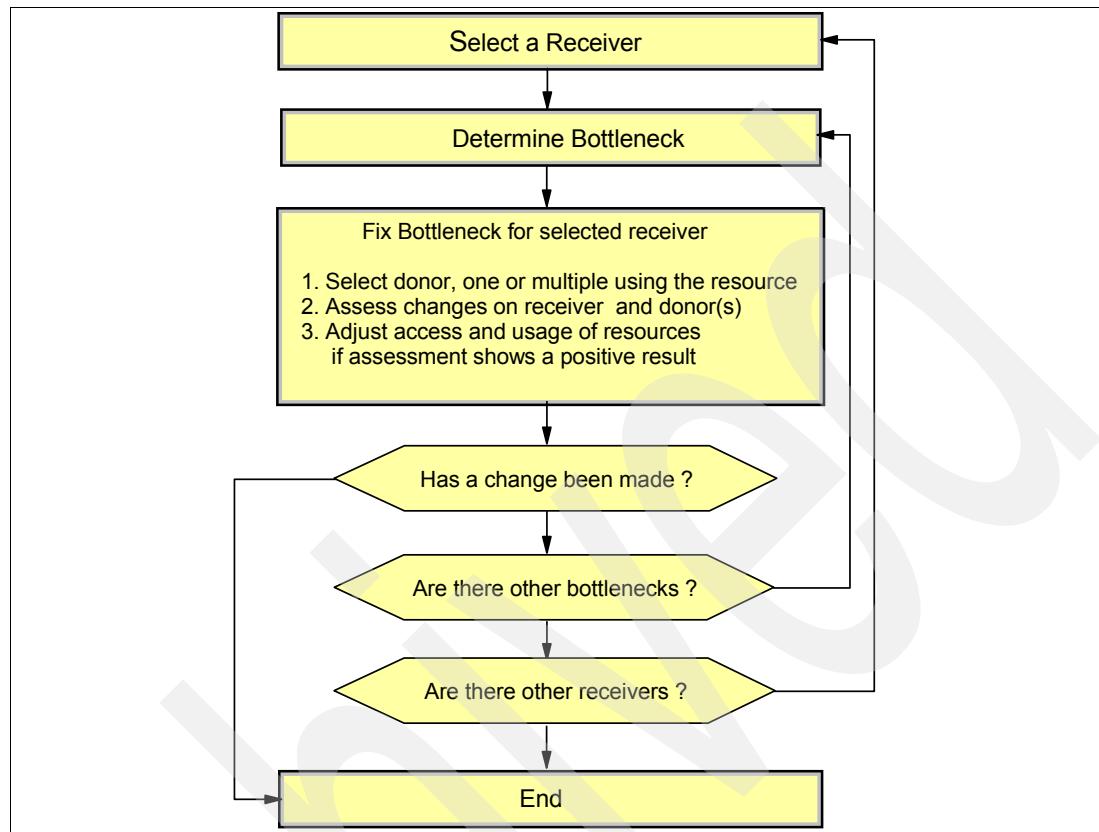


Figure 3-12 WLM policy adjustment routine

WLM policy adjustment routine

Policy adjustment (PA) is how SRM does the following:

- ▶ Selects work to help
- ▶ Selects performance bottlenecks to address
- ▶ Selects which work should donate resource to help other work
- ▶ Assesses whether it is worth it to help work

The purpose of policy adjustment is to meet service class and resource group goals. Policy adjustment is done approximately every 10 seconds.

Every 10 seconds the WLM policy adjustment routine gains control in one z/OS in the sysplex. In a sysplex, the WLM PA routines are not synchronized in time. It summarizes the system state information sampled (every 250 milliseconds) and combines it with real measured data, as the transactions' local average response time. Its major objective is to select a receiver service class period (LPAR) and several donor LPARs (or it could be just one). The receiver will have a raise in a local priority figure. The donors will have a decrease in a local priority figure. The donor service class must be a heavy user of the resource that is delaying the receiver LPAR. This priority is associated with the resource queue causing the highest delay in the receiver transactions during the last 10 seconds.

WLM managing priorities

At this point WLM learns the behavior of the workloads and the system. Based on this information, the PA algorithm plays its main role such as managing priorities to meet service class periods and resource group goals. This is achieved, as mentioned, through:

- ▶ Increasing a receiver's service class period priority, and
- ▶ Decreasing a donor's service class periods priority

To better understand how WLM plays with such priorities, refer to 3.15, "CPU management" on page 100 and 3.21, "I/O management" on page 108.

The priority involved in the exchange applies to the resource causing delays in the LPAR receiver that is causing a PI greater than 1. To have a raise in a set of dispatchable units in one LPAR, you always need to decrease someone else's priority. The major reason for such behavior is to decrease the effect in other transactions (the ones which, at this 10-second interval, are not donors and not receivers). Because all transactions in an LPAR are treated as a whole, all dispatchable units (TCB/SRB) running in the same LPAR in one z/OS have the same priorities (with the exception of the discretionary goal). However, this is *not* true if they run in different z/OS systems.

LPAR receivers

The logic for choosing the LPAR receiver is based on the PIs (local and global) and importance. Only one receiver is managed at a 10-second interval time. The policy adjustment loop (refer to 3.12, "WLM policy adjustment routine" on page 93 for more information) uses the sysplex PI value at the beginning of each 10-second interval to decide whether there is an adjustment it can make to improve the sysplex PI (greater than 1) for the most important unhappy LPAR, starting the search at importance 1. Note that each WLM acts upon its local PI to decrease the sysplex PI (a weighted average of local PIs).

If a LPAR receiver cannot be found (that is, if there are no happier heavy users of the resource causing the delay on the receiver) or if the sysplex PI is less than 1, then the policy adjustment loop is performed a second time, this time focusing on the local LPAR PI (greater than 1 and very important). An adjustment is made, if possible, to help an LPAR from the local system perspective. That is, a LPAR is identified as a receiver when its global PI of an importance 1 service class period is less than 1 ("happy"), but the local PI is greater than 1. This is to compensate for overachievement on one system cancelling out the effects of poor performance on other systems that are processing the same service class periods. If a receiver LPAR cannot be selected, then the same logic is executed at importance level 2, 3, and so on.

Parallel Sysplex clustering

The zSeries Parallel Sysplex cluster contains innovative multisystem data sharing technology. It allows direct, concurrent read and write access to shared data from all processing nodes in the configuration, without sacrificing performance or data integrity.

Each node can concurrently cache shared data in local processor memory through hardware-assisted cluster-wide serialization and coherency controls. As a result, work requests that are associated with a single workload, such as business transactions or data base queries, can be dynamically distributed for parallel execution on nodes in the sysplex cluster based on available processor capacity.

Parallel Sysplex technology builds on and extends the strengths of zSeries e-business servers by linking up to 32 servers with near-linear scalability to create the industry's most powerful commercial processing clustered system. Every server in a Parallel Sysplex cluster has access to all data resources, and every "cloned" application can run on every server.

Using the zSeries “Coupling Technology,” the Parallel Sysplex technology provides a “shared data” clustering technique that permits multisystem data sharing with high performance read and write integrity. This shared data (as opposed to “shared nothing”) approach enables workloads to be dynamically balanced across all servers in the Parallel Sysplex cluster.

This approach allows critical business applications to take advantage of the aggregate capacity of multiple servers to help ensure maximum system throughput and performance during peak processing periods. In the event of a hardware or software outage, either planned or unplanned, workloads can be dynamically redirected to available servers, thereby providing near-continuous application availability.

To summarize, the sysplex PI is used first in the search for unhappy importance 1 service class periods. If no receiver LPAR is selected, then the local PI is used for importance 1 unhappy transactions. Then, this search continues for other importance (2, 3, 4, 5) LPARs. This procedure helps to manage installations where heterogeneous transactions run in the same service class, but on different systems.

The donor LPAR is usually a happy and (PI less than 1) heavy user of a resource causing the most delay in the LPAR donor.

Note: The service classes SYSTEM and SYSSTC do not have an explicit numeric goal and can never be candidates for having their priorities changed.

3.13 WLM system-provided special service classes

- ❑ SYSTEM: DP=255 and IOP=255 for system address spaces
- ❑ SYSSTC: DP=254 and IOP=254 for some system and subsystem address spaces, default for subsystem STC (in Classification Rules)
- ❑ SYSOTHER: goal is Discretionary, default for all subsystem not STC (in Classification Rules)

Figure 3-13 WLM system-provided special service classes

WLM special service classes

A service class has one or more periods with the goal and the importance described by the installation. However, there are system address spaces that, to guarantee adequate system performance, are assigned to specific and predefined service classes:

- ▶ SYSTEM
- ▶ SYSSTCx (x is a number from 1 to 5 or blank)

SYSTEM and SYSSTCx service classes are related to STC address spaces that are either created by the START command or during the IPL process. The creation of these address spaces is through the ASCRE (address space creation) macro. The STC address spaces are assigned to SYSTEM or SYSSTCx, depending on the parameters passed as input to this function and the ones declared at SCHEDEXx Parmlib member.

SYSTEM and SYSSTCx can also be assigned to address spaces or transactions through the classification rules. However, this assignment is not recommended. To prevent the misclassification of system address spaces, WLM in z/OS V1R10 keeps the following address spaces as a SYSTEM service class even if they are differently defined in the policy:

- ▶ MASTER SCHEDULER, WLM, XCFAS, GRS, SMSPDSE, SMSPDSE1, CONSOLE, IEFSCHAS, IXGLOGR, SMF and CATALOG

Service class SYSTEM properties

The following are fixed priorities:

- ▶ Dispatching priority (DP) = 255 (X'FF') and
- ▶ I/O Priority = 255 (X'FF')

There is no goal associated with such a service class, and its DP and IOP never change.

In addition to the previously cited address spaces, the following address spaces use, automatically, the SYSTEM service class:

- ▶ DUMPSEERV, RASP, CONSOLE, IOSAS, SMXC, ANTMMAIN, JESXCF, and ALLOCAS

Service class SYSSTCx properties

The SYSSTCx service classes have fixed priorities:

- ▶ Dispatching Priority 254 (X'FE') and
- ▶ I/O Priority 254 (X'FE')

There is no goal associated with such a service class, and its DP and IOP never changes, as follows:

- ▶ It is the default for the subsystem STC in the classification rules. If you do not define another default, all STC address spaces not filtered through any rule will be assigned to the SYSSTCx service class.

Examples of address spaces using SYSSTCx are: SMS, LLA, JES2AUX, JES3, JES2, VTAM, JES2MON, VLF, APPC, ASCH, RRS, DFMSrmm, and OAM.

It is possible to define service classes SYSSTC1, SYSSTC2, SYSSTC3, SYSSTC4 and SYSSTC5. Up to z/OS 1.9, such service classes have the same properties as SYSSTC.

SYSOTHER service class

SYSOTHER has a discretionary goal. It is the default in classification rules for all non-STCs without a classification rule. Many times transactions are running in such a service class because the classification rules by mistake did not catch them. However, this situation should be avoided because discretionary goals produce good performance only if all the other service classes with numeric goals are happy.

3.14 Dispatching priorities assignment

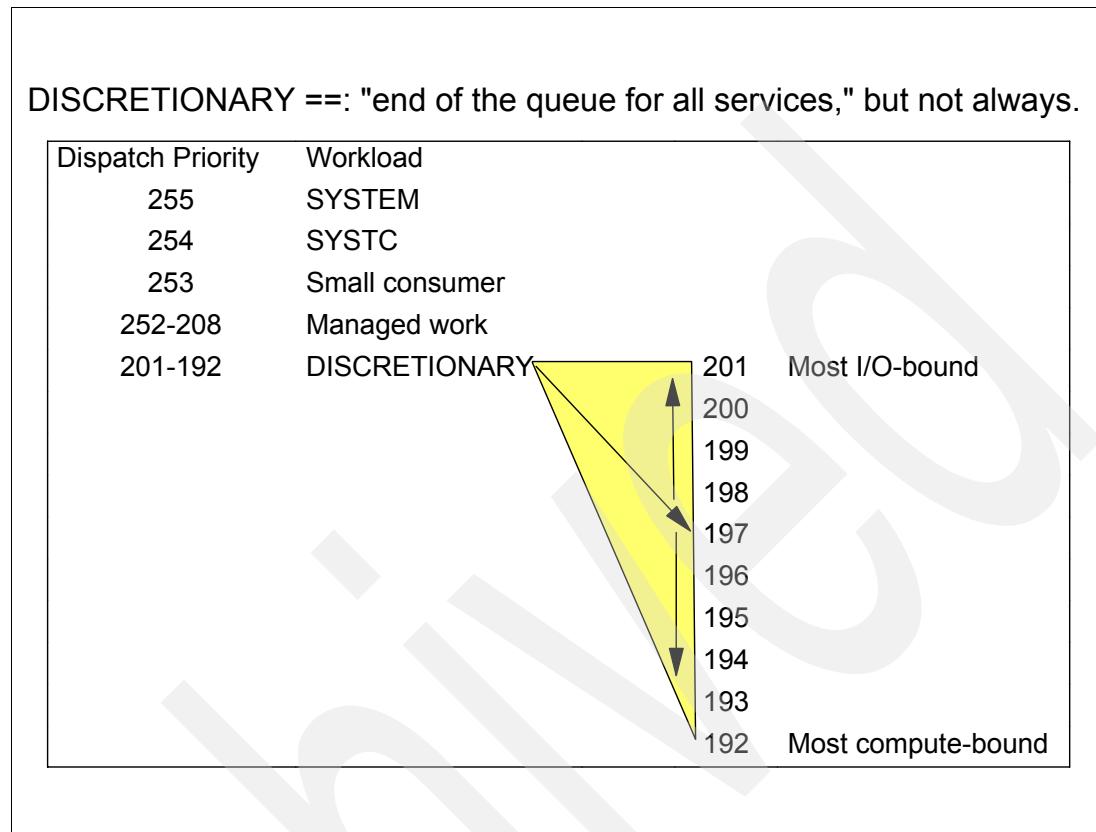


Figure 3-14 Dispatching priorities assignment

Dispatching priorities assignment

A dispatch priority is a number from 0 to 255 associated with a dispatchable unit (TCBs and SRBs) related with the execution of a transaction. It is placed in an address space or in an enclave control block. It indicates the priority of these dispatchable units for getting PUs (CPs, zAAPs or zIIPs) service. In goal mode, dispatching priorities are not assigned by the installation, but by WLM based on the service class period goals.

SYSSTC service class

In goal mode, initiators run in service class SYSSTC with a fixed dispatching priority of 254 and there was no way to adjust or change the dispatching priority. The work running under the initiator is assigned its installation-specified service class during SYSEVENT JOBSELCT processing. If there are CPU-intensive exits that run prior to job select processing, the work will run at dispatching priority 254 and this can potentially impact high importance work.

Figure 3-14 lists the range of dispatch priorities and illustrates how WLM assigns them to the transaction in the system.

The dispatch priority of 255 is reserved to keep the system running and to provide system transaction preferential access to the CPU. All system transactions should run in the predefined service class SYSTEM, which always gets a dispatch priority of 255.

Note: The SPM qualifier in the WLM classification rules helps to automatically define this transaction in the proper service classes, although the system address spaces place themselves at proper service class (SYSTEM or SYSSTCx) through an option in the macro ASCRE.

Other dispatch priorities

One special dispatch priority is used for dispatchable units in service class periods that consume very little PU (CPU, zAAP, or zIIP) service and are I/O bound, because it is beneficial for the whole system to allow such dispatchable units to use a little CPU for starting a new I/O operation (creating parallelism between CPU and I/O, which is good for throughput) and enter voluntary wait. To accomplish this, WLM gives this transaction a dispatch priority just below the two system-related dispatch priorities of 253.

Below this starts the range of dispatch priorities (from 208 to 252) that are dynamically assigned to user-defined service classes periods (not included in this range are the discretionary ones). These are all service classes with an importance of 1 to 5. All dispatchable units of the same service class period have the same base dispatching priority (with the exception of the mean-time-to-wait group for discretionary transaction) in the same z/OS. The dispatching order of all dispatchable units with the same dispatching priority in the same or in different service class periods are naturally rotated.

The range from 202 to 207 is not used and the range from 192 to 201 is used for the dispatchable units of discretionary transactions. All discretionary transactions are managed by a mean-time-to-wait (MTW) algorithm which, simplified, gives a higher dispatch priority for transactions using I/O and a lower dispatch priority to transactions demanding significant PU. The last used dispatch priority is 191. It is assigned to quiesced transactions running in a non-swappable address space. That also means that quiesced transactions can run if the PU is not busy, and all other transactions are in wait state or active in other PUs.

The interesting range is from 208 to 252. WLM assigns these dispatch priorities to the user-defined service classes periods that have a goal. The assignment is based on the PU consumption, PU demand, and goal achievement of the transaction.

3.12, “WLM policy adjustment routine” on page 93 explains that WLM determines, through the performance index, which service class periods meet their goals and which ones miss their goals. In the next step, WLM starts to look for a receiver by starting with the most important service class period with the worst PI. Then it determines the resource (based on the largest delay numbers) for which an adjustment is required.

For PU, that means that the service class must show a high number of PU delays. Now WLM looks for donors. Based on the captured data, WLM knows how much PU is consumed by each service class period and how much PU is consumed by all transactions at the same dispatch priority. From the PU delays and the average time that a transaction used the PU, it is possible to calculate the demand that the transaction has for the PU.

To assign a new dispatch priority to a receiver service class period, WLM logically moves the service class period to a higher dispatch priority and then projects the change of PU consumption, demand, and delays for all service class periods at a lower and at the same new dispatch priority of the transaction being moved up. If that is not sufficient, WLM looks in the next step to determine whether some service class periods can run with a lower dispatch priority and does the same calculations. Refer to 3.15, “CPU management” on page 100 for more details. The algorithm continues until a positive result can be projected, meaning that the performance of the receiver gets better and no service class periods with a higher importance is badly affected, or until WLM concludes that no change is possible.

3.15 CPU management

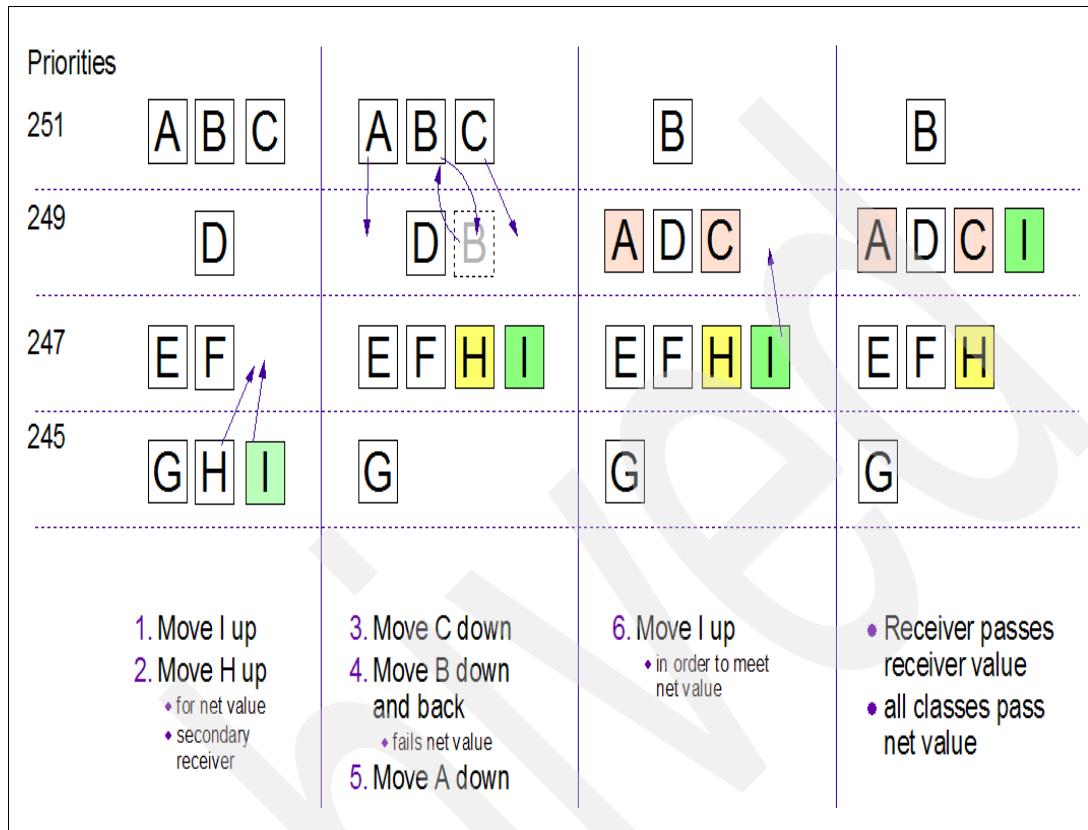


Figure 3-15 WLM CPU, zAAP, and zIIP dispatching priority adjustment

CPU management

Access to the PUs (CPs, zAAPs, and zIIPs) is controlled through dispatching priorities. z/OS maintains three queues of ready transaction units, which are sorted in priority order. The z/OS dispatcher component selects the next ready dispatchable unit (TCB or SRB) going through such queues. This section explains how these queues are managed by WLM. WLM assigns dispatch priorities to dispatchable units associated with transactions running in service classes periods, based on the following factors:

- ▶ The need of the service class period
 - Goal achievement, expressed by performance index
 - CPU consumption
- ▶ The potential impact of a change
 - Basically simulates the movement of service classes period dispatchable units to higher (receivers) and lower (donors) dispatching priorities.
 - Forecast, through calculation, the change in service consumption. Forecast, through calculation, the change in goal achievement.
 - Changes dispatch priorities if it is beneficial (through a calculated net value) for the higher importance transaction.

An example of the CPU management algorithm

This example explains how WLM works with dispatching priorities between donors and receivers. Figure 3-15 illustrates the CPU management algorithm. The example assumes

nine service class periods, and the policy adjustment algorithm selected service class period I as a potential receiver of the CPU service.

In the first step, it moves I to a higher dispatch priority. Together with I, there are two other service class periods, G and H, at the same dispatch priority. After the first move up for I, the calculations show that H also has to move up because otherwise, no overall value of the move for service class I can be projected; in other words, H would be degraded.

This projection is named “net value,” meaning that the overall change does not harm higher importance transactions and is still beneficial for the receiver. But this move does not show enough receiver value, meaning that the change does not help I enough to improve its performance. In the next step, WLM tries to move transactions with higher dispatch priorities to lower dispatch priorities. While the move down for C is acceptable, the move down for B violates the net value test.

Finally, the move down of A passes the net value test, but again there is not sufficient receiver value for I. In the third step, WLM again attempts to move the receiver up. In this case the move for service class I to dispatch priority 249 gives enough receiver value for I and all other classes to pass the net value tests. In this case, the A and C service class periods are the donors to service class I, which is the receiver.

Now WLM concludes that the change is meaningful and adjusts the dispatch priorities of the transaction. Note that if no valuable change could be projected, all intermediate calculations would have been set back and no change would have occurred.

Also observe that, together with service class I, service class H was moved to a higher dispatch priority. This became necessary because the first move showed that H would have been degraded too much. Ultimately H is below I, but now competes with service classes E and F at the same dispatch priority and therefore has better and sufficient access to the CPU.

The final picture has changed to some extent. The moves are calculated based on the current and the expected CPU consumption, which is derived from the CPU demand of the service classes and the amount of CPU that is available at each dispatch priority level.

This example also demonstrates why WLM only executes one change per policy adjustment interval. One change does not mean that only the resource access for one service class period has been changed—it means that the *access* to a resource for a set of service class periods is changed to help one service class period.

Note: Keep in mind that goals generally should be based on SLA (that is, response time-based). For LPARs out of SLA, you need to observe transaction behavior. Do not define unrealistic goals; this is especially important for the execution velocity type of goal.

3.16 Reasons for CPU delays

HARDCOPY RMF V1R2 Processor Delays										Line 1 of 56		
Command ==>												
0Samples: 119	System: MVS1		Date: 11/03/05		Time: 13.06.00		Range: 120		Sec			
0	Service DLY USG Appl EAppl ----- Holding Job(s) -----											
Jobname	CX	Class	%	%	%	%	%	Name	%	Name	%	Name
0SECICS2	SO	SCL004	14	39	38.1	38.1	5	SECICS5	5	SECICS3	4	SECICS1
BBI1SS	SO	SCLMONIT	12	6	2.7	2.7	6	SECICS2	5	SECICS3	3	SECICS6
SECICS1	SO	SCL004	6	20	16.0	16.0	3	SECICS2	2	SECICS5	2	SECICS1
SECICS9	SO	SCL004	6	20	16.7	16.7	3	SECICS3	3	SECICS9	2	SECICS2
SUQ2297	T	SCLTSOP	5	6	1.8	1.8	3	SECICS2	2	SECICS3	2	SECICS6
SECICS4	SO	SCL004	4	9	6.5	6.5	3	SECICS2	2	SECICS3	1	SECICS5
TCPPIP	SO	SCL001	4	5	2.6	2.6	2	DBP1MSTR	2	SECICS6	2	SECICS2
SECICS3	SO	SCL004	3	21	20.8	20.8	2	SECICS2	2	SECICS9	2	DBP1MSTR
SECICS5	SO	SCL004	3	18	11.3	11.3	3	SECICS2	2	SECICS3	1	SECICS4
SECICS6	SO	SCL004	3	16	11.1	11.1	1	SECICS5	1	SECICS3	1	APPC
DBP1DBM1	S	SCL002	2	9	6.9	6.9	1	SECICS3	1	RMFGAT	1	SECICS6
BBPAS1	SO	SCLMONIT	2	3	1.6	1.6	2	SECICS6	1	SECICS7	1	SECICS2
APPC	S	SYSTC	2	3	0.1	0.1	2	APPC	1	SECICS1	1	SECICS9
VTAM	S	SCL001	2	2	0.7	0.7	2	APPC	1	SECICS1	1	SECICS9
NETV1	SO	SCLMONIT	2	0	0.1	5.0	1	SECICS5	1	SECICS3	1	SECICS2
NPMPPROC	SO	SCLMONIT	2	0	0.3	0.3	2	SECICS3	1	SECICS5	1	SECICS2
DBP1MSTR	S	SCL002	1	8	4.3	4.3	1	BBI1SS				

Figure 3-16 RMF monitor III PROC report

Reasons for CPU delays

Figure 3-16 shows a RMF Processor Delay report. You see address space SECICS5 causing delays (5%) in SECICS2 (suffering a total of 14% CPU delay), which is an address space running in SCL004 service class. There are several causes that justify holding DUs causing PUs (CPU, zAAP, zIIP) delay in others:

- ▶ Holding is more critical to business and WLM associated to it a higher dispatch priority.
- ▶ Holding is less critical to business and WLM temporarily associated a higher dispatch priority because:
 - It is I/O bound, then almost no harm to the other.
 - Delayed ones are happy in relation to their defined goals, or the holding ones are promoted because they own an enqueue or a lock for which there is contention. If you disagree, then make the service class period goal more difficult to be accomplished and increase the importance. Also, you may activate the CPU protection (CPU Critical) WLM feature or define a minimum value in a resource group. Refer to Figure 3-18 on page 105 for more information about this topic.
- ▶ Holding is less critical to the business and WLM associated it with a lower dispatch priority. However, because of z/OS reduced preemption (a low dispatching priority task is preempted by a high one only after getting a minimum amount of CPU time), the holder is delaying others. However, these delayed figures should be very small (not larger than 3%). If you disagree, you may avoid this by using WLM resource group capping.

3.17 CPU Critical option

```
Service-Class Notes Options Help
-----
Create a Service Class                                     Row 1 to 2 of 2
Command ==> _____
Service Class Name . . . . . WASSC      (Required)
Description . . . . . WebSphere sc
Workload Name . . . . . ONLINES    (name or ?)
Base Resource Group . . . . . (name or ?)
Cpu Critical . . . . . YES        (YES or NO)

Specify BASE GOAL information. Action Codes: I=Insert new period,
E>Edit period, D>Delete period.

---Period--- -----Goal-----
Action # Duration Imp. Description
_____
= 1           3     80% complete within 00:00:01.000
***** Bottom of data *****
```

Figure 3-17 Service class panel with CPU Critical option YES

CPU Critical option

CPU Critical is another WLM CPU management function defined in the service class definition in the WLM policy (Figure 3-20 on page 107). It protects long-term CPU Critical transactions, ensuring that less important transactions generally have a lower dispatch priority.

As discussed in 3.16, “Reasons for CPU delays” on page 102, it is possible that transactions that are in service classes period with easier goals and lower importance get higher dispatch priority than transactions with a high importance.

As a result, a service class period with low importance, easier goals, and low CPU service consumption can obtain the highest available dispatch priority in the system. Note the following points:

- ▶ This is not a problem as long as the transaction behaves well.
- ▶ However, it can hurt higher important transactions when the transaction starts to use greater amounts of CPU in an unpredictable manner. This often happened in CICS when, after a low activity period (for example, lunch time), users begin to log on and experience elongated response time. After some policy adjustment, WLM gives CICS the proper dispatching priority and things run fine again. However, the purpose of the CPU Critical feature is to eliminate this problem.

CPU Critical usage

The CPU Critical feature protects transactions by not allowing transactions with lower business importance (easier goal and low importance) to get a higher dispatch priority. This protection can be valuable for transactions that are extremely CPU-sensitive, such as certain CICS and IMS transactions.

The transaction that needs protection is flagged. The result is that:

- ▶ All transactions with higher or equal importance can still compete freely for CPU resources with the protected service class.
- ▶ All transactions with lower importance will generally have a lower dispatch priority.

This can help in situations where low importance transactions may negatively impact high importance transactions (for a short but visible period).

The service class protected by CPU Critical must have just one period.

Be careful not to overuse this feature, because otherwise you do not leave enough capability for WLM to efficiently balance the available CPU resources across the service classes to meet goals. WLM tries to meet not only the goals of the highest importance transactions, but also to meet those of lower importance transactions, if the resource permits it.

3.18 CPU Critical and resource groups

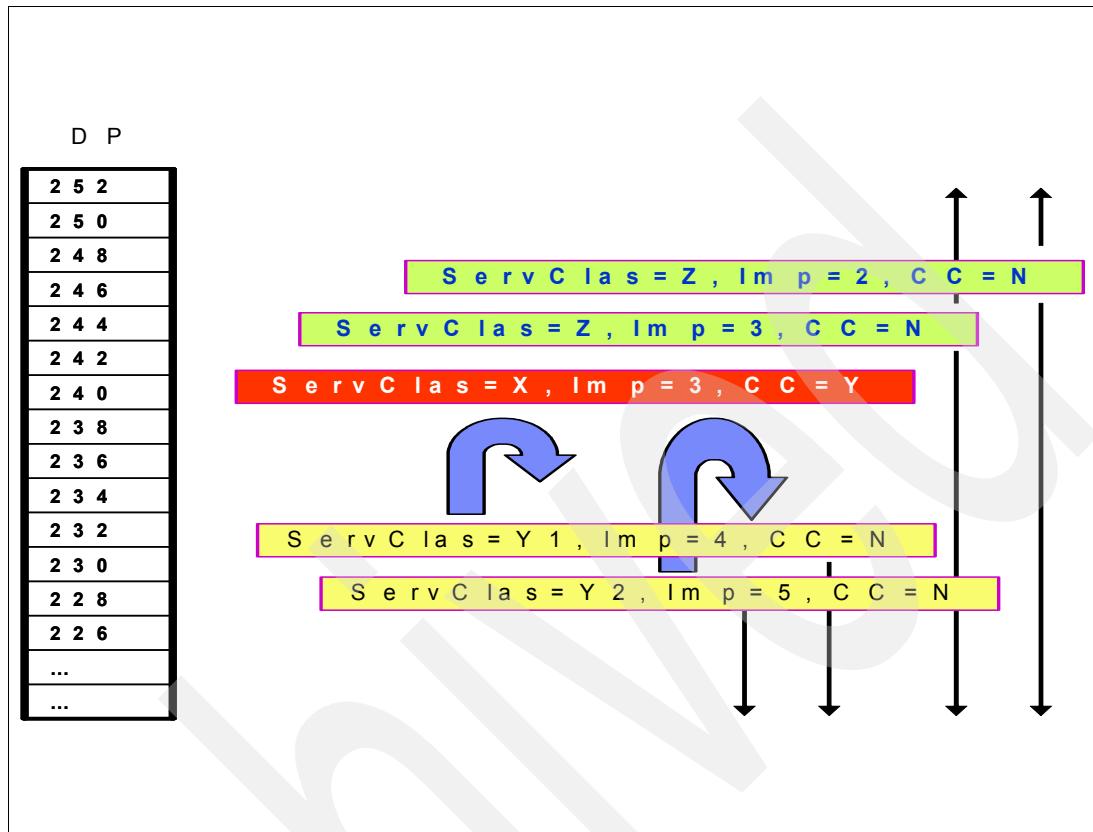


Figure 3-18 CPU Critical in action

CPU Critical in action

Figure 3-18 shows the effect of the CPU Critical feature: service class X (Imp 3) has been flagged as CPU Critical (CC=Yes). All service classes having an importance higher than or equal to it compete for CPU resource access (service class Z periods in the figure) in a normal way. However, service classes Y1 (Imp 4) and Y2 (Imp 5) will never be allowed to have a dispatch priority higher than service class X or any service class of importance 3, 2, or 1.

This feature should be used for high business importance workload. CICS/IMS production workloads are good candidates for this feature because their activity can be subject to fluctuations.

Resource group CPU protection

You can use a Resource Group to guarantee a minimum CPU capacity to the service classes when a transaction in the group is missing its goals. The referred service class must point to this resource group in WLM policy to be protected. This option is called Resource Group CPU protection.

Also, the resource group can be used through a maximum value to limit (capping) the amount of CPU capacity available to some service classes. Capping is used in situations in which you want to deny the access of CPU cycles to one or more service classes. Refer to 4.16, “Resource group capping” on page 154 for more information.

3.19 Storage Critical option

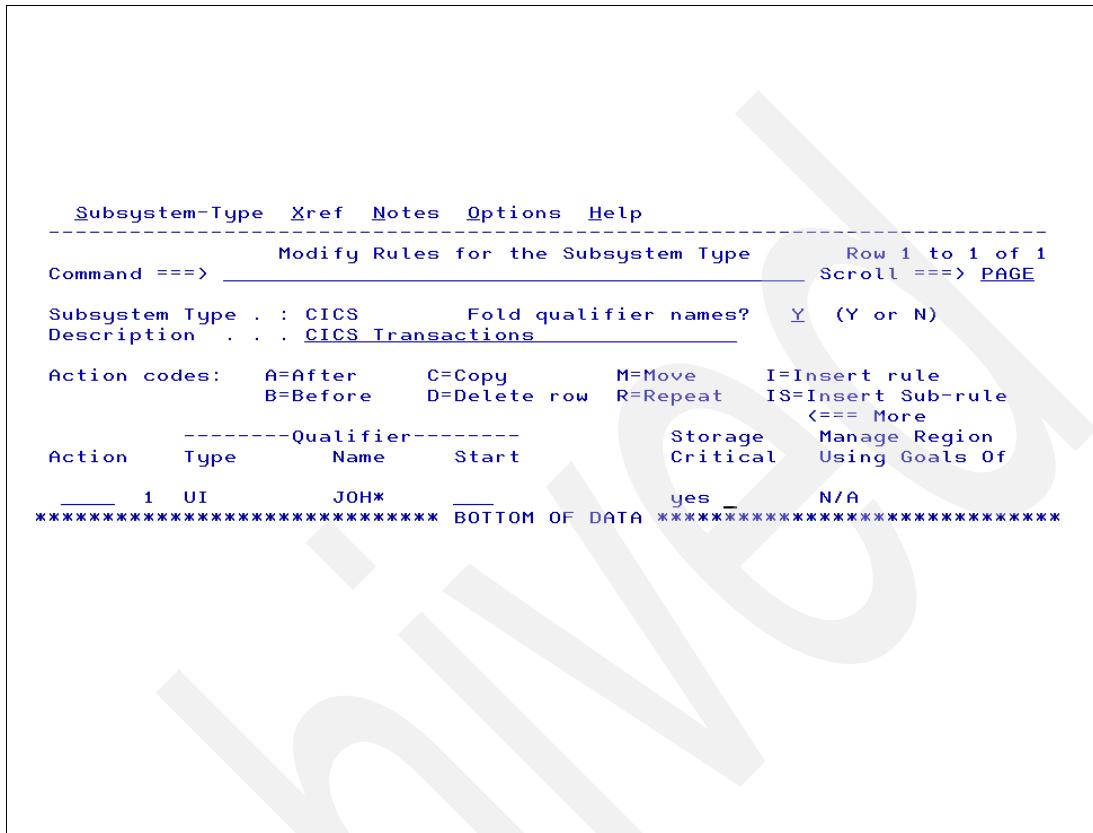


Figure 3-19 Storage Critical option

Storage Critical option

You assign long-term storage protection to critical transactions by using the Storage Critical option in the Classification Rules panel (shifting right using PF11) at the WLM policy (Figure 3-19). With this option, WLM restricts storage donations (stealing) to other address space transactions based on the LRU algorithm.

This option can be useful for address space transactions that need to retain virtual pages in main storage during long periods of inactivity because it cannot afford paging delays when it becomes active again. Without such a function, the LRU stealing algorithm steals the pages least referenced, if main storage enters contention. With long-term storage protection assigned, this transaction address space loses main storage (by stealing) only to transactions of equal or greater importance that need the storage to meet performance goals.

Storage of protected service classes is only taken when the z/OS runs in short on main storage conditions.

3.20 Storage Critical in action

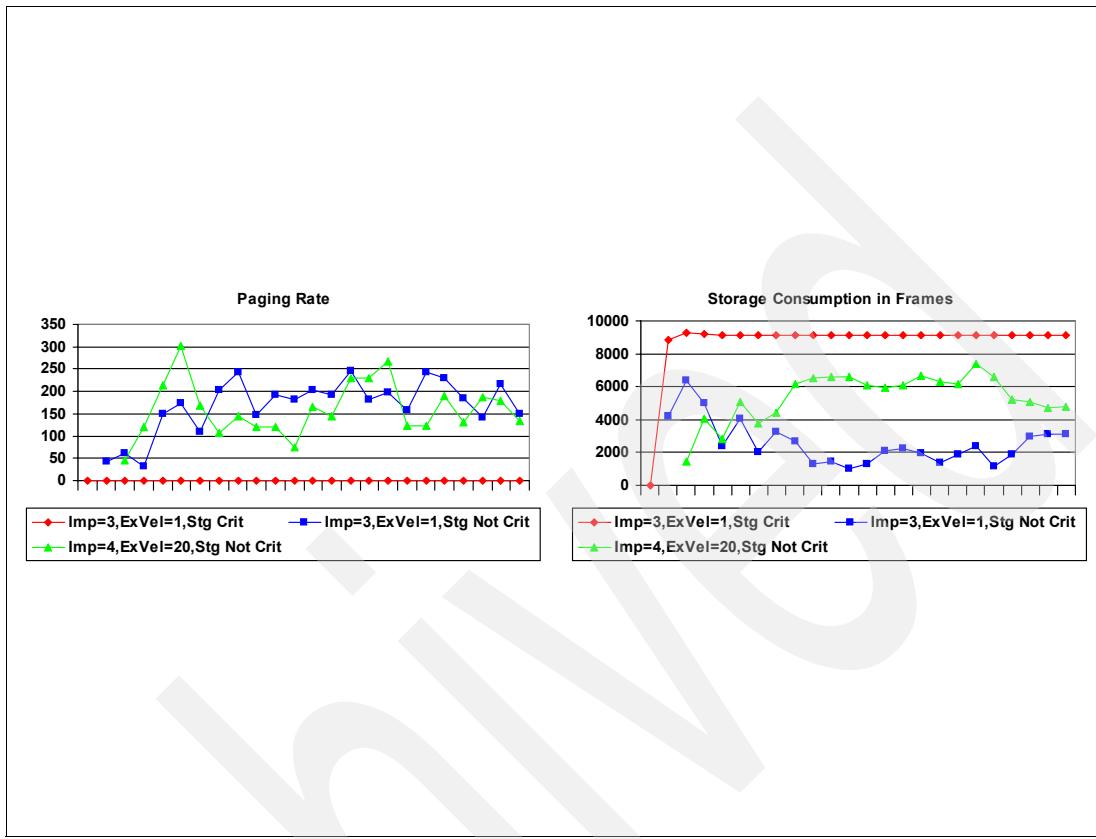


Figure 3-20 Storage Critical in action

Storage Critical in action

The leftmost graph in Figure 3-20 shows that the Imp=3 service class with the Storage Critical attribute (listed at the bottom) is protected against stealing from the Imp=3 and Imp=4 service classes, which are without the Storage Critical attribute. No stealing implies a no paging rate. The right-most graph shows the amount of main storage allocated to the IMP 3 Storage Critical service class to guarantee a zero page fault rate.

Other characteristics are that storage remains protected even if no paging activity is going on anymore. That is important for online regions (CICS/IMS) that do not show much activity overnight.

IRLM (the DB2 lock manager address space) has an erratic paging activity and can be protected against page stealing by using the Storage Critical option.

3.21 I/O management

- ❑ I/O Priority management
 - I/O queues
 - DASD I/O priority
 - Tape I/O priority
 - Channel subsystem I/O priority queueing
- ❑ Dynamic Parallel Access Volumes (PAV)

Figure 3-21 I/O management

I/O management

I/O management in WLM has several functions, such as I/O priority management (to dynamically control I/O priorities) and Dynamic Parallel Access Volume (PAV).

I/O priority is a number used to place I/O requests in I/O queues. I/O priority can be optionally managed by WLM if I/O priority management is set to Yes in the WLM policy. This option is recommended. It is independent from the CPU dispatching priority. WLM uses eight different I/O priority values to classify I/O operations depending on the transactions requiring such I/O.

I/O queues

There are three types of I/O queues where I/O priority management can apply:

- ▶ IOS UCB queue - This is a local z/OS queue, because it is per device and all requests come from the same z/OS. The queue occurs when there are no available UCBs to start the I/O operation.
- ▶ Channel subsystem queue - This is a global CEC queue because the requests come from all logical partitions existing in that CEC. The queue occurs when there is no full I/O path (channel, port switch, controller, and device) that is totally free. Thus, there could be competition between all the logical partitions to access a particular device.
- ▶ Control unit queue - This is a global queue because the requests come from all logical partitions of all connected CECs. IBM control units starting with 2105 are able to use the I/O priority passed in the I/O request and determined by WLM to order such queue.

Highlights of I/O priority management

An I/O priority assessment between two or more service class periods takes place in WLM Policy Adjustment when these service class periods use the same set (or subset) of devices.

Thus, if one important service class period misses its goal (it is a candidate to be a receiver) and policy adjustment finds out that I/O delay is the biggest problem, then WLM tries to find other service class periods (candidates to be donors) that use the same set (or a subset) of the devices for the service class period that is missing the goals. Then the usual assessment logic starts to determine whether a change in I/O priority is possible between them.

Each I/O request is performed by or on behalf of a z/OS dispatchable unit, either a TCB or an SRB. The dispatchable units are associated with:

- ▶ Address spaces
- ▶ Enclaves

Each address space-oriented transaction and enclave-oriented transaction is associated with a service class period, which becomes the focal point of the management algorithms. For the purposes of managing I/O, WLM controls a small fixed number of I/O priority levels.

As a reminder, the general formula of the response time is:

$$\text{Response_Time} = \text{Service_Time} + \text{Queue_Time}$$

This section explains how to determine where WLM can use priorities to manage the Queue_Time (QT) in a DASD subsystem. Refer to 3.22, “I/O priority management” on page 110 for more information about this topic.

Dynamic Parallel Access Volume

Parallel Access Volume (PAV) allows the same z/OS to execute multiple I/O operations concurrently against the same 3390 volume. There are three ways to implement PAV devices in your installation, depending on the assignment process: by using static; WLM dynamic alias management; or HyperPav. This is discussed in more detail in 3.24, “WLM dynamic alias PAV management” on page 114.

WLM dynamic alias management belongs to WLM I/O management activities.

To implement PAV, IOS introduces the concept of *alias addresses*. Instead of one UCB per logical volume, an MVS host can now use several UCBs for the same logical volume. Apart from the conventional base UCB, alias UCBs can be defined and used by z/OS to issue I/Os in parallel to the same logical volume device.

Currently, the Enterprise Storage Server® (ESS) allows for concurrent data transfer operations to or from the same volume on the same system using the optional feature Parallel Access Volume (PAV). With ESS, alias device numbers can be defined to represent one physical device, which allows for multiple I/O operations to be started at one time. However, the z/OS operating system does not allow more than one I/O operation at a time to the same device and queues additional I/O operations to the physical device.

3.22 I/O priority management

- ❑ I/O priorities for DASD and tape devices are managed in the range of X'F8' through X'FF', where:
 - X'FF' is reserved for high priority system address spaces (service class SYSTEM)
 - X'FE' is reserved for started tasks (service class SYSSTC)
 - X'F8' is reserved for address spaces with a discretionary goal (service class SYSOTHER)
 - The range of priorities between X'F9' and X'FD' is dynamically assigned to I/Os coming from address spaces and enclaves, as determined by goal enforcement algorithms (policy adjustment routine)

Figure 3-22 DASD and tapes I/O priorities

DASD and tape I/O priorities

I/O priorities for DASD and tape devices are managed in the range of X'F8' through X'FF', as shown in Figure 3-22. As you can see by the WLM design, just five I/O priorities (from F9 to FD) is enough to prioritize I/Os coming from dispatchable units associated with transactions running in service classes not SYSTEM and not SYSSTC.

Each service class period maps to an I/O priority level. This mapping is dynamically adjusted based upon how each service class period is meeting its goals and how the I/O contributes to this success or failure. When IOS receives an I/O request and places the I/O request on the UCB queue for the target device (all UCBs are busy), the priority of the request is used to place the request in priority sequence with respect to other pending requests for the same device.

I/O priority management

As with any WLM algorithm, I/O priority management has a sysplex scope. The goals of I/O Priority management support are:

- ▶ To separate I/O scheduling priority from CPU dispatching priority.
- ▶ To manage DASD I/O contention at the sysplex level.
- ▶ To provide algorithms for assigning I/O priorities (at the three I/O queues) to I/Os coming from transactions of various service class periods, based upon information gathered

through sampling techniques and data obtained from the IOS component. It is a donor receiver function when the major delay is the I/O delay.

- ▶ To collect information and distribute the resulting I/O priority adjustments across systems in a sysplex, using existing WLM cross-system communications (XCF) services.
- ▶ For each I/O request, IOS sets the request priority based upon a value established by WLM for the requesting dispatchable unit.

The following changes have been introduced in the I/O process:

- ▶ IOS queues requests on the UCB queue by I/O priority.
- ▶ The I/O priority of an enclave or an address space is adjusted by WLM. Neither the business transaction nor the WLM administrator has control over I/O priority.

I/O Priority management decisions

To manage I/O priorities, information is needed about the delays of each DASD device. When WLM wants to increase a service class period I/O priority, uses the donor and receiver logic WLM policy adjustment 10-second routine (refer to 3.12, “WLM policy adjustment routine” on page 93). Therefore, WLM must be able to determine the following:

- ▶ Whether a service class period can significantly improve its goal achievement by raising its I/O priority (a potential receiver)
- ▶ Whether lowering the I/O priority of a service class period (a potential donor) can improve the achievement of the receiver

WLM needs two types of information to make decisions about I/O priority management:

- ▶ Device service time information

The device service time is the sum of device connect time and device disconnect time for each I/O request, as measured by the I/O subsystem.

WLM can make decisions by comparing the level of device usage across the range of I/O priorities, and projecting the amount of device usage that may occur if I/O priorities are altered.

- ▶ Device wait time (delay) information

WLM can make decisions by comparing the level of device delay across the range of I/O priorities, and projecting the amount of delay that may occur if I/O priorities are altered.

The device delay time is the sum of the time each request was delayed by IOS (IOS queue time) and the time delayed in the I/O subsystem (pending time).

Note: Channel measurement data gives the device connect and disconnect time and the device pending time (including channel busy, controller busy, shared device busy). The IOSQ time is a sampled value. The disconnect time is not taken into account by WLM when calculating velocity% goal.

Channel subsystem I/O priority queuing

Channel subsystem I/O priority queuing (CSS IOQP) extends I/O priority queuing by allowing WLM prioritization of I/O requests within the SAP queues by logical partition and within a logical partition. It allows I/O prioritization through the Service Aid Processor (SAP) queue. Table 3-1 on page 112 lists the priority settings. Note that the SAP I/O priorities numeric values are not the same as those used for the UCB queue and the DASD controller queue.

Table 3-1 Priority settings by type of transaction

Priority	Type of transaction
FF	System transaction
FE	Importance 1 and 2 missing goals
FD	Importance 1 and 2 missing goals
F9-FC	Meeting goals - adjusted by ratio of connect time to elapsed time
F8	Discretionary

This is a part of the Intelligent Resource Director (IRD) feature available on zSeries, z9, and z10 machines in z/Architecture mode. Refer to 3.26, "Intelligent Resource Director" on page 118 for more details,

3.23 Parallel Access Volume

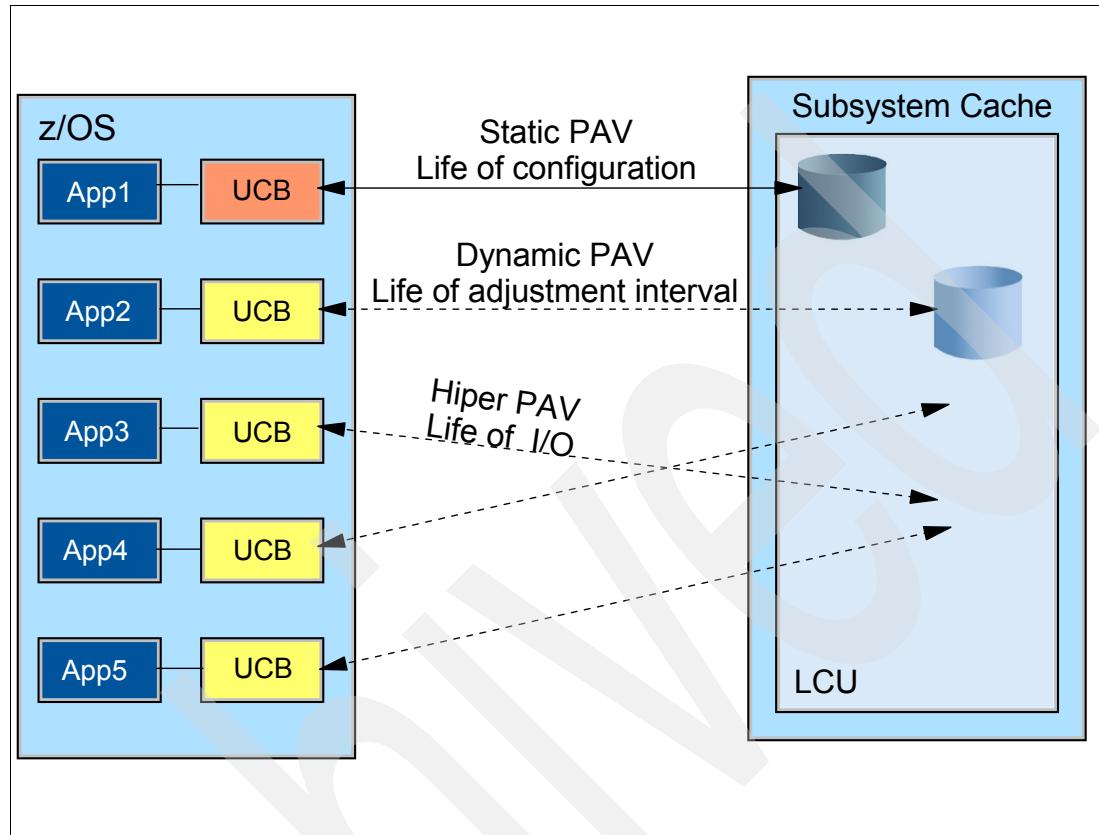


Figure 3-23 PAV modes

Parallel Access Volume

Parallel Access Volume (PAV) implies the possibility of having concurrent I/Os to the same 3390 device coming from the same z/OS. When the concurrent I/Os originate in several z/OS systems, this parallelism is called *multiple allegiance*.

Your installation can manage PAV devices in static, WLM dynamic, or HyperPav modes. If static PAV is used, then there is no change in device alias assignment unless an alias is deleted or added using the DS8000® console. Figure 3-23 shows the PAV modes.

This section describes the current implementation of the WLM algorithms for *dynamic* alias management with the DS8000 in some detail. WLM can be used to dynamically manage the assignment of alias devices to base devices and thus ensure that the concurrency in I/O processing provided by PAV is available where it is most needed. However, in general the HyperPav implementation is superior to WLM dynamic alias management.

3.24 WLM dynamic alias PAV management

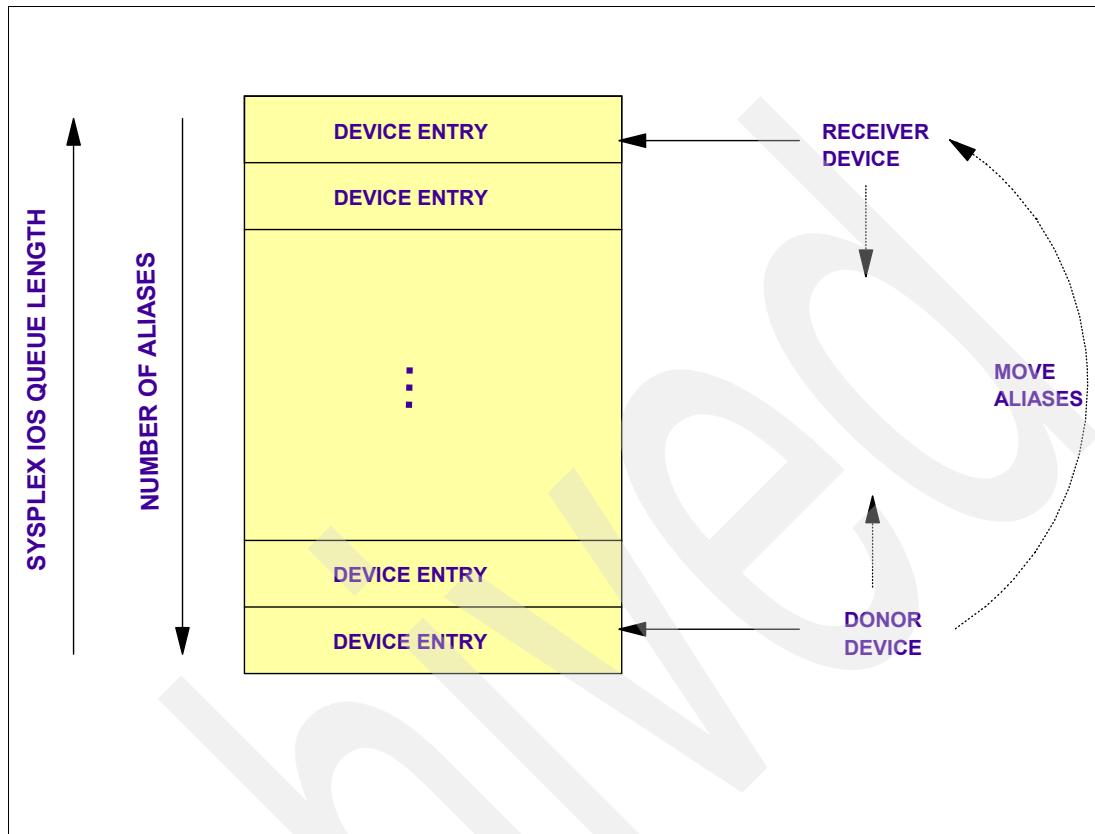


Figure 3-24 PAV data table for a sysplex

WLM dynamic alias PAV management

The WLM function has two algorithms:

- ▶ Efficiency adjustment simply tries to balance UCB aliases across devices, as long as no service class is negatively impacted. It spreads out unbound (UCBs originally not connect to devices) aliases. It runs every 60 seconds, and is executed by the WLM Resource Adjustment routine.
- ▶ PAV goal adjustment watches the goal achievement of the service classes in the systems and takes corrective action (if more UCB aliases would help) by moving alias UCBs for a donor device to a receiver device; refer to Figure 3-24 for more information about this topic. It runs once every 30 seconds, and is executed by the WLM Policy Adjustment routine.

To implement this function, you must specify YES for Dynamic Alias Management in the WLM policy and WLMPAV=YES in each involved device in HCD.

The donor and the receiver UCB bases must be in the same LCU.

WLM alias PAV specification in HCD

There may be instances where a storage administrator may want to specify that a range of device addresses be excluded from Dynamic Alias Management. This can be achieved by using the HCD WLMPAV specification.

WLMPAV=YES | NO can be specified for base or alias devices. For bound aliases, the WLMPAV specification for base device applies to base or alias devices. Specifying WLMPAV=NO means that WLM will not assign PAV aliases to this base or remove them. For unbound aliases, the WLMPAV specification for the alias device applies.

This specification is local per z/OS; a device can be WLMPAV=YES in one z/OS, and WLMPAV=NO in another z/OS.

Specifying WLMPAV=NO for alias (device type 3390A) devices has no effect for a bound alias, but protects an unbound alias from being a preferential candidate for Dynamic Alias Management. This may be desirable in several instances:

- ▶ Some devices in a shared LCU are normally online to some system images, but offline to others.
- ▶ When an installation has several sysplex with shared access to a DS8000 LCU.

The DS8000 does not distinguish between static and dynamic PAVs; there are only PAVs.

PAV dynamic alias management for paging devices

Your system availability may be impacted when certain events, such as an SVC dump, cause a burst of paging activity that locks out or slows down page fault resolution for your critical workloads. New, combined ASM/WLM exploitation of parallel access volumes (PAVs) on the IBM DS8000 allows the system to prevent page-outs from blocking page-in operations.

Enhancements in z/OS V1R3 allow auxiliary storage manager (ASM) to state the minimum number of PAV dynamic aliases needed for a paging device. The more page data sets you have on the volume, the more alias devices will be set aside. A minimum number of aliases will be preserved by the system even in cases where aliases are being moved to reflect changes in workload. You do not need to define any of this; simply enable the volume for dynamic alias management for your paging devices.

Note: All systems in your sysplex must be on a z/OS V1R3 or above level for this support.

3.25 HyperPAV

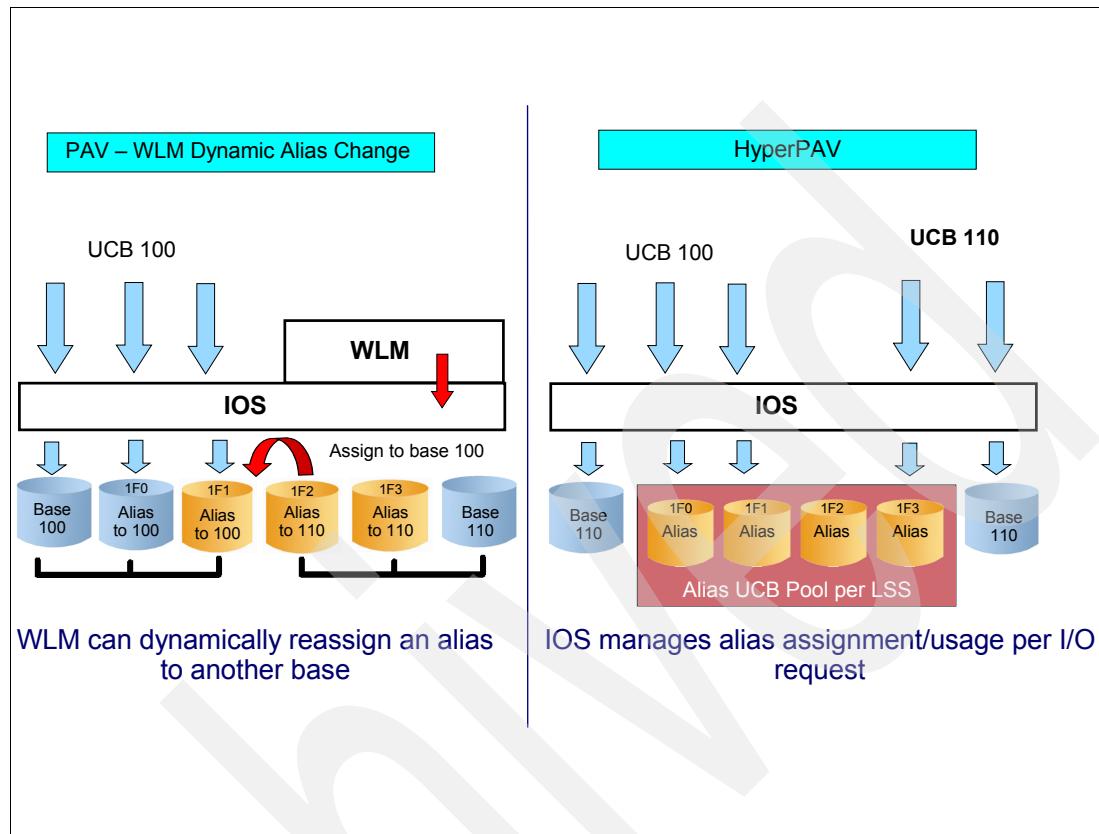


Figure 3-25 HyperPAV logic

HyperPAV logic

HyperPAV is more efficient than WLM dynamic PAV and eventually will probably replace it. HyperPAV does not use WLM intelligence to distribute the UCB aliases through the UCB base at the 3390 devices. The work is done at IOS level, when the I/O operation is suppose to start. The logic follows:

- ▶ IOS uses a shared pool of LSS free UCB aliases as defined in HCD. LSS stands for logical subsystem and it is the DS8000 name for logical control unit (a set of 256 devices).
- ▶ As each I/O operation is requested, if the base volume is busy with another I/O:
 - IOS selects a free UCB alias from the pool, quickly binds the alias device to the base device and starts the I/O (through SSCH instruction).
 - When the I/O completes (I/O interrupt), the alias device is returned to the free alias pool.
- ▶ The I/O operation is queued only if the mentioned LSS free pool is empty. RMF has data covering such situation.

With HyperPAV, WLM is no longer involved in managing alias addresses. For each I/O, an alias address can be picked from a pool of alias addresses within the same LCU. This capability also allows different HyperPAV hosts to use one alias to access different bases. This reduces the number of alias addresses required to support a set of bases in a System z environment with no latency in targeting an alias to a base. This functionality is also designed

to enable applications to achieve better performance than is possible with the original PAV feature alone, while also using the same or fewer operating system resources.

Benefits of HyperPAV

HyperPAV has been designed to provide an even more efficient parallel access volume (PAV) function. When implementing larger volumes, it provides a way to scale I/O rates without the need for additional PAV alias definitions. HyperPAV exploits FICON architecture to reduce overhead, improve addressing efficiencies, and provide storage capacity and performance improvements, as follows:

- ▶ More dynamic assignment of PAV aliases improves efficiency.
- ▶ The number of PAV aliases needed might be reduced, taking fewer from the 64 K device limitation and leaving more storage for capacity use.

3.26 Intelligent Resource Director

□ LPAR CPU management

- WLM manages an LPAR cluster (set of z/OSs in the same CEC, all of them in goal mode and in the same Parallel Sysplex) to provide the processor resources necessary for the work to meet its goals, based on business importance. The processor resources are obtained by dynamically changing the weight and the number of logical CPUs.

□ Dynamic channel path management

- Ability to move channels to DASD control units as needed, through switches.

□ Channel subsystem priority queuing

- Extension of I/O priority queuing by allowing WLM prioritization of I/O requests within the SAP queues by logical partition and within a logical partition.

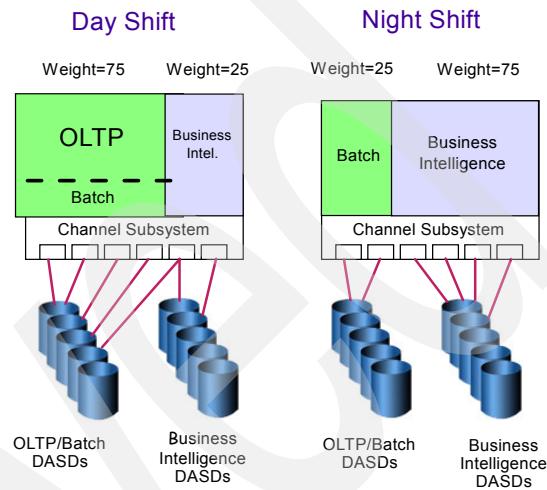


Figure 3-26 Intelligent Resource Director features

Intelligent Resource Director

Intelligent Resource Director (IRD) is not actually a product or a system component. Instead, it is comprised of three separate but mutually supportive functions implemented by actions on:

- ▶ z/OS (WLM, IOS)
- ▶ LPAR code in zSeries, z9 and z10
- ▶ SAP

Figure 3-26 shows the features of IRD.

IRD functions

These IRD functions are:

- ▶ LPAR CPU Management
 - WLM Vary logical CPU management
 - WLM Weight management
- ▶ Dynamic Channel Management (ESCON channels with ESCON Directors only)
- ▶ Channel subsystem priority queuing (CSS IOPQ)

3.27 Moving resources through IRD

- ❑ Earlier, MVS customers had just one image
- ❑ Now, they create many images using LPARS due to:
 - Need for isolation
 - Need to easily install new workloads
 - Need for continuous operation
- ❑ With Parallel Sysplex, many "data sharing" images are in place under LPARs and this allows dynamic workload balancing - that is, sending the transaction to where the resources are more available
- ❑ However, some images are still out of data sharing (creating affinities)
 - ❑ In this case, IRD takes the resource to where the transactions are running

Figure 3-27 Moving resources through IRD

Moving resources through IRD

Originally a CEC was designed to host only one copy of operating system code (single image). However, several current issues (such as the need for: isolation; easy installation of new workloads; continuous operation) create many logical partitions images under the LPAR hypervisor in the same CEC. Because sometimes, each system is unique in itself with its own data, and because program problems (such as complexity, unbalancing, and lack of availability) arise from having many LPs, the solution was to allow data sharing among the LPs through the implementation of Coupling Facilities. Now any transaction can run anywhere, and with dynamic load balancing algorithms you may balance the load in LPs in the same and different CECs. This means that you take the transaction to where the hardware is more available. All the systems may be clones because they can reach the same data and programs.

However, for certain types of data it is difficult to implement data sharing. In these cases, the unbalanced behavior is not completely resolved. IRD introduces the opposite idea of dynamically taking the hardware (CPUs and channels) to the z/OS LP where it is most needed, thereby resolving the imbalance problem. In other words, we can move the hardware resource (CPU and channels) power to where it is needed.

WLM and IRD

With the function CPU Management, WLM with IRD can change dynamically the number of logical PUs in a logical partition accordingly with the load and the LPAR overhead. Also, WLM can take LP weight from one LP (donor) and deliver to other (receiver) depending on the

distribution of the workload and goals not being honored. To summarize, z/OS with WLM allows you to drive a processor at 100% while still providing acceptable response times for your critical applications. IRD amplifies this advantage by helping you make sure that all those resources are being utilized by the right workloads, even if the workloads exist in different LPs.

Dynamic Channel Path Management

DCM in general (and FICON DCM, for that matter) allows z/OS to manage channel paths (FICON and ESCON) dynamically. The client must identify the channels and control units to be managed.

By assigning channels to the pool of managed channels, the system is able to respond to peaks in a control unit's demands for I/O channel bandwidth. In addition, this reduces the complexity of defining the I/O configuration, because the client is no longer required to define a configuration that will adequately address any variations in workload. This allows them to specify a much looser configuration definition.

DCM can provide improved performance by dynamically moving the available channel bandwidth to where it is most needed. Prior to DCM, you had to manually balance your available channels across your I/O devices, trying to provide sufficient paths to handle the average load on every controller. This means that at any one time, some controllers probably have more I/O paths available than they need, while other controllers possibly have too few.

Another benefit of Dynamic Channel Path Management is that you can be less focused on different rules of thumb regarding how busy you should run your channels for every channel or control unit type you have installed. Instead, you simply monitor for signs of channel over-utilization (high channel utilization combined with high pending times).

Because Dynamic Channel Path Management can dynamically move I/O paths to the LCUs that are experiencing channel delays, you can reduce the CU and channel-level capacity planning and balancing activity that was necessary prior to Dynamic Channel Path Management.

Using DCM, you are only required to define a minimum of one non-managed path and up to seven managed paths to each controller (although a realistic minimum of at least two non-managed paths is recommended), with Dynamic Channel Path Management taking responsibility for adding additional paths as required, and ensuring that the paths meet the objectives of availability and performance.

3.28 LPAR cluster

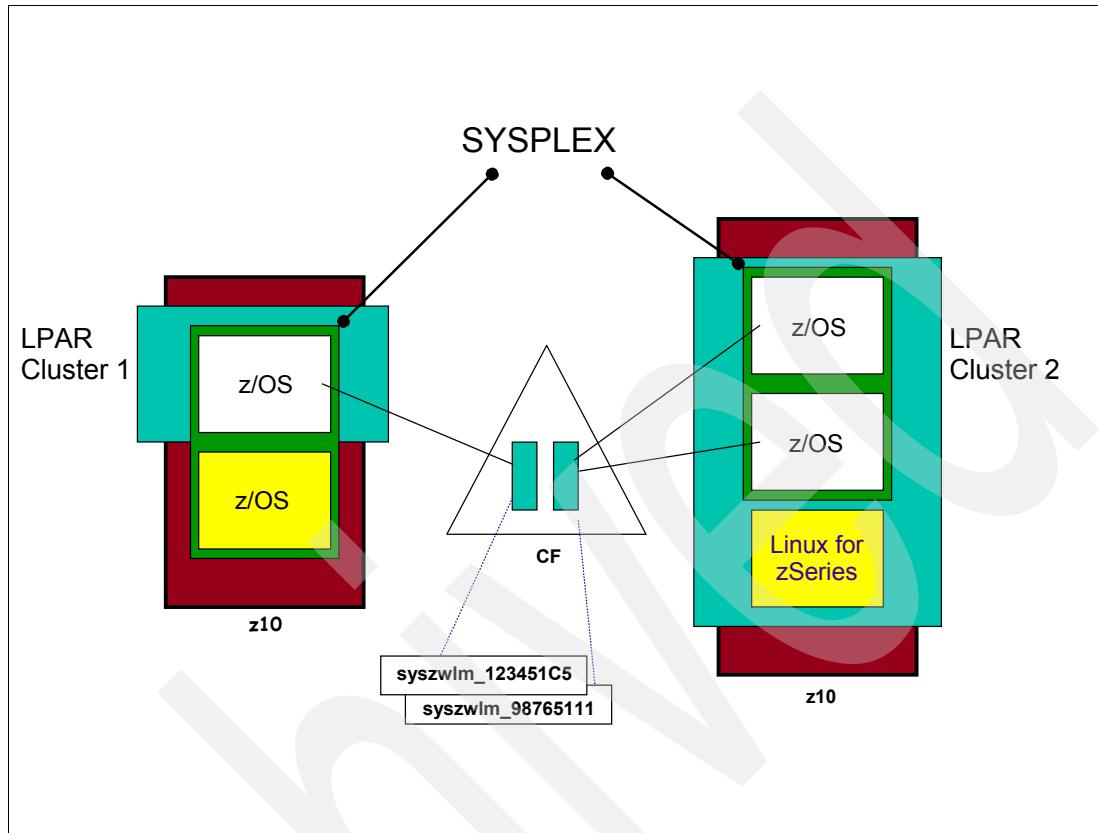


Figure 3-28 LPAR cluster

LPAR cluster

IRD uses the concept of an LPAR cluster, which consists of the subset of z/OS systems (you also may have LPs with Linux, with or without z/VM®) that are running as LPs on the same zSeries (or z9, or z10) server. The z/OS images must be in the same Parallel Sysplex.

IRD uses facilities in z/OS Workload Manager (WLM), Parallel Sysplex, SAP (which is the processor unit in charge of starting an I/O operation) and LPARs to help you derive greater value from your investment.

An extensive discussion about IRD is beyond the scope of this book. Instead, this book describes how to create a WLM policy that transacts properly with IRD. All IRD functions and setups are fully explained in the Redbooks publication *z/OS Intelligent Resource Director*, SG24-5952.

3.29 RMF LPAR cluster report

L P A R C L U S T E R R E P O R T													
z/OS V1R5 SYSTEM ID SC69 DATE 11/07/2004 INTERVAL 14.59													
RPT VERSION V1R5 RMF TIME 16.30.00 CYCLE 1.000 sec													
----- WEIGHTING STATISTICS -----										PROCESSOR STATISTICS			
CLUSTER	PARTITION	SYSTEM	INIT	MIN	MAX	Avg	MIN %	MAX %	NUMBER	DEFIN	ACTUAL	LBUSY	PBUSY
WTSCPLX1	A0A	SC69	10	1	999	19	0.0	0.0	16	2.0	99.30	11.03	
	A01	SC55	10	1	999	1	100	0.0	2	2.0	80.48	8.94	
	A02	SC54	10	10	2	2	95.05	10.56					
	A03	SC49	185	185	2	2	99.91	11.10					
	A07	SC52	10	10	2	2	95.05	10.56					
	A08	SC48	10	10	2	2	95.05	10.56					
	A09	SC47	185	185	2	1	99.63	5.53					
	TOTAL			420					28		664.5	68.30	

Figure 3-29 RMF LPAR cluster report

RMF LPAR cluster report

The RMF LPAR cluster report, as shown in Figure 3-29, displays IRD performance data. Here, we activate IRD WLM Weight management function.

In the LPAR Cluster WTSCPLX1 there are 7 LPs. In HMC, the LPs A0A and A01 are defined with a minimum Weight of 1 and a maximum of 999. As you can see, the Weight (WGT) values for A0A and A01 are changed from the initial values, due to IRD Vary Weight Management action. For LP A01, it is equal to the minimum 1. For LP A0A, it is 19. The amount of weight lost by the donor (A01) is equal to the amount received by the receiver (A0A).

The reason why IRD raised the weight of A0A is because an important service class running in A0A was unhappy ($PI > 1.0$), mainly because of a delay for CPU. The weight was taken from A01, where there are less important service classes that may be allowed to suffer.

Also, the IRD function WLM Vary CPU Management was activated for LPs A0A and A01 due to the decimal point in the 2.0 in the ACTUAL field. However, it seems that the value was kept constant along the RMF interval, meaning that WLM found the correct figure for the interval. More logical CPUs implies more capacity but at same time more LPAR overhead, and WLM balances both.

3.30 Managing CICS and IMS/DC WLM goals

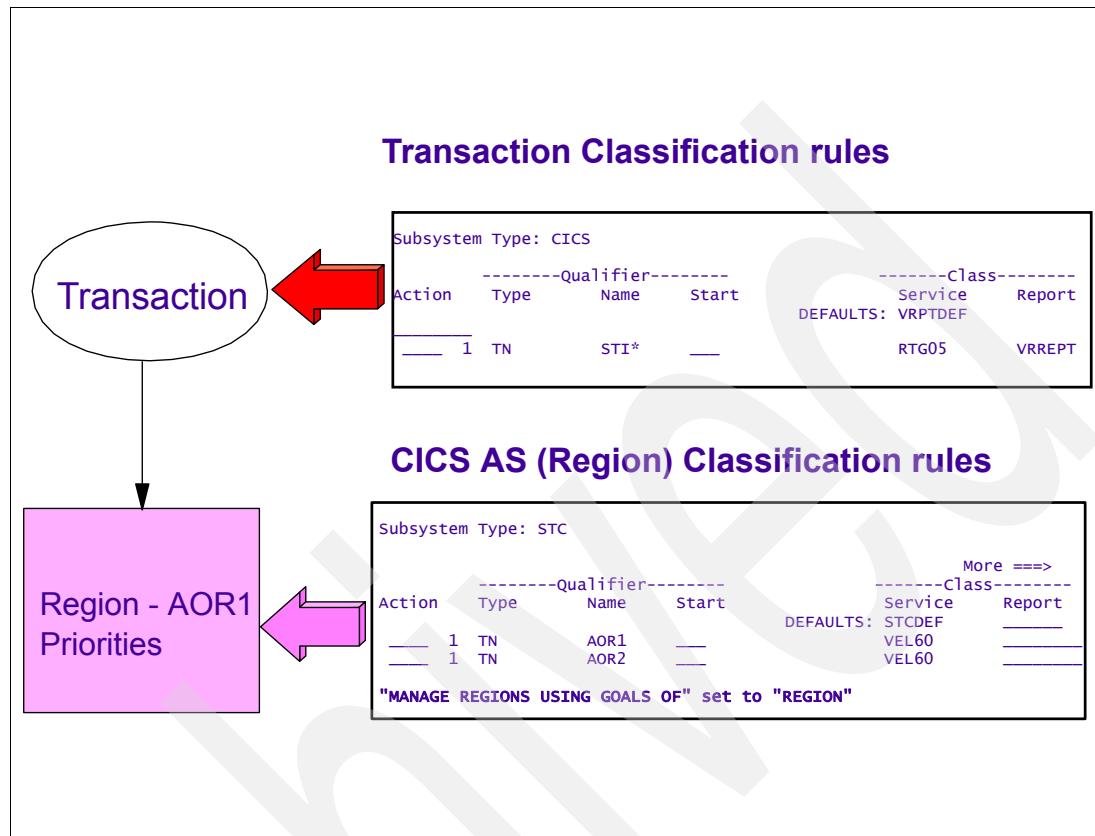


Figure 3-30 Response time and Region goals

Managing CICS and IMS/DC WLM goals

Note that CICS and IMS/DC do *not* implement enclaves for their transactions. With enclaves, the transactions can have unique (separated from the address space) WLM priorities and account information for each one. These priorities and accounts are independent from those located in the address space basis (in the ASCB control block). The major consequence of this is that WLM is unaware of the existence and whereabouts of CICS and IMS/DC transactions.

To solve that, WLM developed a set of programming interfaces that allow transaction managers such as CICS and IMS/DC to identify their transactions to WLM. This helps WLM trace the transactions through the transaction manager address space and to identify the transaction response time in z/OS. WLM developed various techniques to recognize transactions on the system and to manage them independently from, or together with, the address space where the programs run to process them.

In CICS or IMS/DC, the installation may be defined in the policy, as shown in Figure 3-30:

- ▶ Execution Velocity% goal declared in subsystem STC (Classification Rules) for the ASs (regions) executing the transactions. Based on that, WLM calculates these ASs priorities. In this case, WLM is not informed about transactions. This method is called Region.

- Or, you can use the transaction response time (average or percentile) as a goal declared in subsystem CICS (or IMS). In this case, the ASs priorities are decided by the PIs and importance of the running transactions.

Then, in CICS and IMS/DC, the priorities (I/O and CPU) are in an address space level and in each address space you may have numerous transactions belonging to different service classes. This method is called Transaction.

The transactions have the response time goal, but the address space structure has the priorities. In addition, a transaction during its execution might span multiple address spaces on z/OS through cross-memory. To manage this scenario, WLM must recognize the transaction flow and, based on this, manage the address spaces to meet user expectations. Refer to 3.31, "CICS/IMS address space server topology" on page 125 to learn how WLM performs this task.

It is also possible to either manage some CICS/IMS regions (address spaces) to velocity goals or to manage some others as response time. Then, installations could select certain regions (for example, regions of less importance, regions processing conversational transactions, or regions with very low activity) and have WLM manage them to the velocity goal, while the more important ones with high activity are best managed by WLM according to the transaction service classes response time goals.

The installation can force one of the methods (Region or Transaction) for certain CICS/IMS address spaces through the classification rules associated with certain transactions. If you declare, for example, Region, then even if the transaction arrives with a response time goal, the method used is Region.

The process works like this: when the CICS address space is started, WLM uses the Velocity goal as declared in the subsystem STC. (We recommend defining an aggressive goal to start CICS quickly.) When the first transaction arrives, WLM verifies whether the transaction has a subsystem CICS rule pointing to a service class response time goal.

If there is no such rule, then WLM keeps using the Velocity of the address space. However, if there is such a rule, then the address spaces priorities are decided by the PIs and importance of the running transactions. The point is to always define a region execution velocity goal for a CICS production (for a fast start), even if you plan to define a subsystem CICS response time.

3.31 CICS/IMS address space server topology

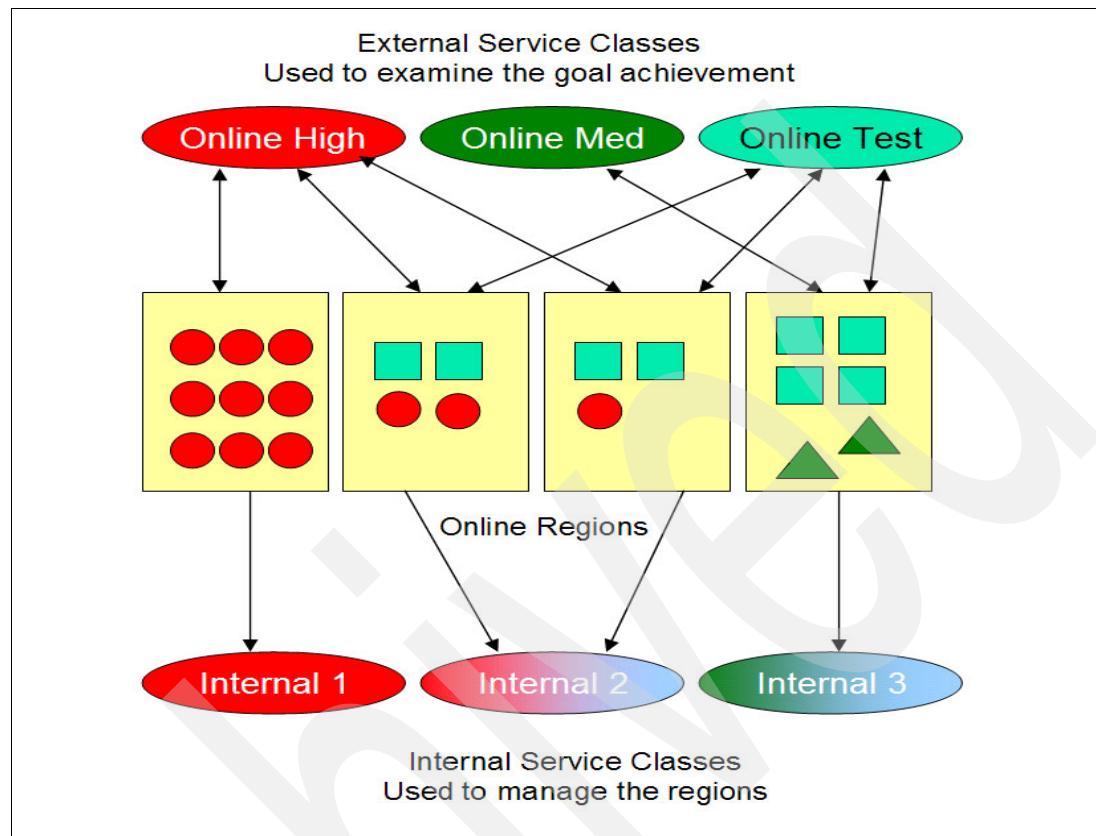


Figure 3-31 CICS/IMS server address space topology

CICS/IMS address space server topology

The transaction service class is specified through classification rules in the subsystem CICS and has a response time goal.

WLM creates a picture of the address space topology, including the server address spaces and the service classes of the running transactions. All this information is passed by CICS. It uses this topology to manage the server address spaces priorities based on the goal achievement of the most important and most unhappy service classes. The “free ride” phenomena may occur here, where in the same heterogeneous address space, other transactions enjoy high figures in priorities caused by very important and very unhappy other service class transactions.

To understand the distribution of transaction through server ASs, WLM derives a concept known as “internal service classes.” One internal service class is a set of regions executing the same set of external service class transactions.

Figure 3-31 depicts the server topology. It shows three external service classes (the ones defined in WLM policy) for CICS or IMS/DC transaction and four registered server regions (address spaces) that process the transactions.

WLM needs to identify the number of internal service classes. One region processes only requests for service class Online_High (circles). As a result, one internal service class is created, which is used to manage this region.

Two regions process transactions from Online_High and Online_Test (squares) only. The second internal service class is used for these two server regions, and the last region is associated with the third internal service class (Online_Med transactions are represented by triangles).

Assume that the external Service Class Online_High misses its goals. Through the topology, WLM determines that three server regions process work requests of this Service Class, which are associated with two internal Service Classes. Then it calculates how much each of the internal Service Classes contributes to the goal achievement of the external Service Class.

Further assume that WLM finds out that the server regions of the internal class 2 need help (they are suffering WLM measured delays). Therefore, WLM helps the server regions of this Service Class to improve the goal achievement of the external Service Class Online High.

Archived

3.32 CICS transaction type of goal



```
Subsystem-Type Xref Notes Options Help
-----
Modify Rules for the Subsystem Type Row 16 to 16 of 16
Command ==> Scroll ==> PAGE
-----
Subsystem Type . : STC      Fold qualifier names? Y (Y or N)
Description . . . Started Tasks
-----
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      CICS      NO        TRANSACTION
***** BOTTOM OF DATA *****

```

Figure 3-32 Classification rule for CICS address space defining transaction type of goal

Working with classification rules

To move from velocity to response time goals:

- ▶ Create one or more single-period service classes with real time goals.
- ▶ Create classification rules in the CICS subsystem in the STC or JES subsystem section (depending on whether CICS runs as an STC or a batch job).
- ▶ In the rule (or rules) that name the CICS region address spaces, mark them as “Management Region Using Goals Of: TRANSACTION,” as shown in Figure 3-32.

For JES and STC work only, you can also fill in the “Manage Regions Using Goals Of:” field, with a value of either TRANSACTION, in which case the region will be managed to the transaction response time goals, or REGION, in which case the region is managed to the goal of the service class assigned to the region. Note that for ASCH, CICS, IMS, and OMVS work, the “Manage Regions Using Goals Of:” field will read N/A.

Archived

WLM miscellaneous functions

This chapter describes a large set of WLM functions not directly connected with goal achievement. It contains the following items:

- ▶ Scheduling environment
- ▶ Dynamic workload balancing
- ▶ Server address space management
- ▶ Application environment
- ▶ Capping
- ▶ Just in time capacity
- ▶ Capacity provisioning
- ▶ WLM management of zIIP and zAAP
- ▶ Dispatchable unit promotion
- ▶ Buffer pool management
- ▶ WLM console commands
- ▶ WLM Q&A
- ▶ Common mistakes in WLM policy scheduling environment

4.1 Scheduling environment

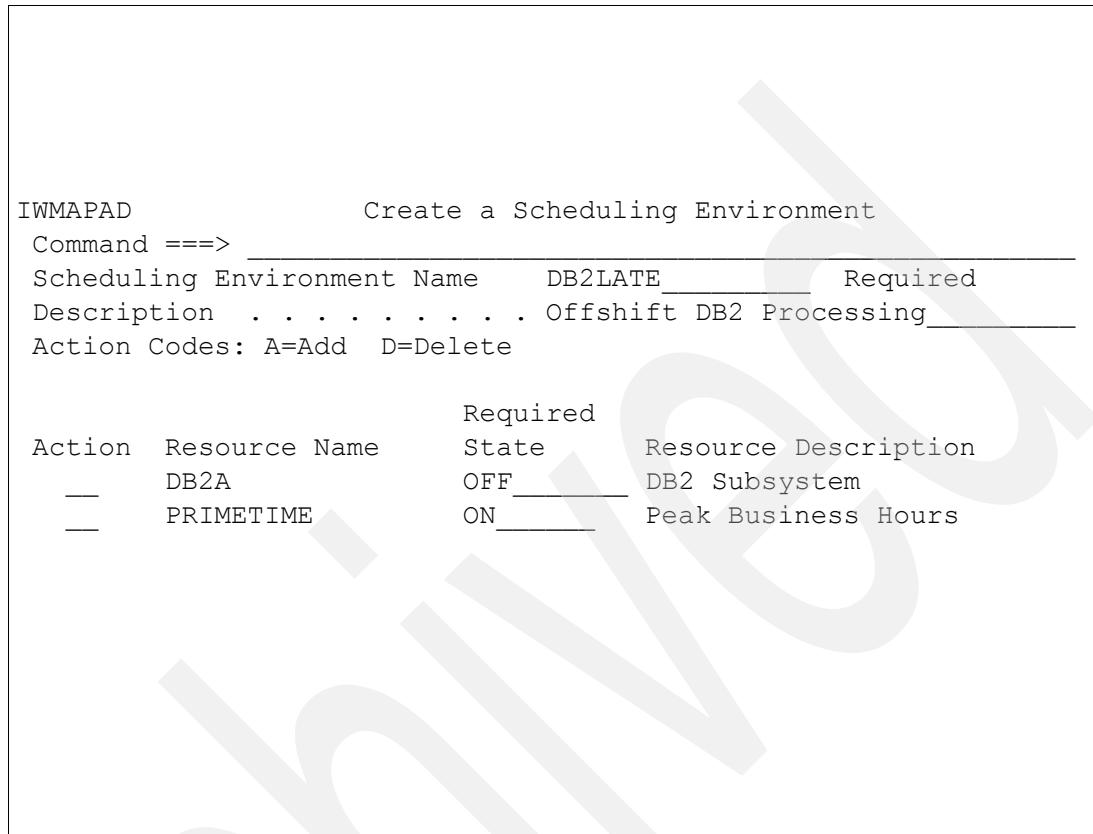


Figure 4-1 Scheduling environment definition panel

Scheduling environment

The use of the scheduling environment facility is optional. WLM can also be in charge of routing batch jobs to z/OS systems that have the resources that will be needed by such jobs. Thus, the function is available for the JES2 multi-access spool facility and JES3. A scheduling environment lists resource requirements, which ensures that transactions are sent to systems that have those resources. This is particularly important in sysplex installations where the z/OS systems are not identical.

The defined resources can be physical entities (such as hardware features or subsystem instance), software products, or intangible entities such as certain periods of time (transactioning hours, off-shift hours, and so on).

As illustrated in Figure 4-1, the scheduling environment (DB2LATE) lists the resources (through any resource name) and their required state, ON or OFF existent in all the z/OS systems. In the example shown in the figure, DB2A is OFF and PRIMETIME is ON. The resources are also named scheduling elements. Each arriving transaction (mainly a batch job) has a keyword in JCL informing the required scheduling environment name to select the z/OS system that contains the desired scheduling elements options described in the referred scheduling environment name.

Through a **MODIFY WLM** command, the installation may set ON or OFF the scheduling elements in a particular z/OS.

4.2 Scheduling environment implementation

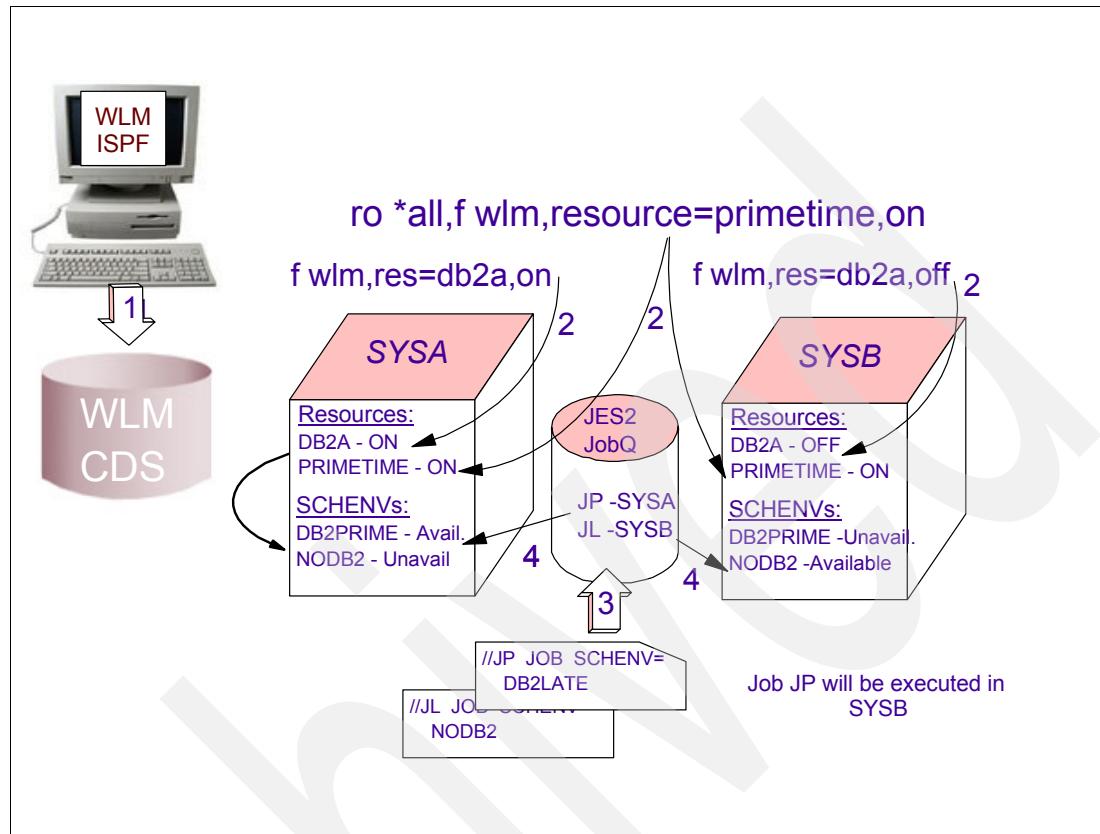


Figure 4-2 Resource affinity scheduling

Scheduling environment

A scheduling environment is a combination of sysplex systems, database managers, devices, and languages capable of executing an application. In other words, it is a set of resource elements with specific values (ON or OFF) set. In Figure 4-2, the scheduling environment DB2LATE formed by the resource elements states: DB2A OFF and PRIMETIME ON.

Resource affinity scheduling

Resource affinity scheduling is a WLM facility exploited by both JES2 and JES3 transaction managers. The WLM resource affinity scheduling function extends the capability of JES2/JES3 workload scheduling components by enhancing their scheduling mechanism, which means selecting the right system for the job. Scheduling environments are defined in the WLM ISPF application in a policy that is stored in the WLM CDS, as shown in Step 1 of Figure 4-2.

As mentioned, scheduling environments help to ensure that units of work are sent to systems that have the appropriate resources to handle them. They list resource names along with their required states. Resources can represent actual physical entities, such as a database or a peripheral device, or they can represent intangible qualities such as a certain period of time (like second shift or weekend).

These resources are listed in the scheduling environment according to whether they must be set to ON or set to OFF. A unit of work can be assigned to a specific system only when all of

the required states are satisfied. This function is commonly referred to as resource affinity scheduling.

Resource affinity scheduling allows an installation to do the following:

- ▶ Define resource elements in the sysplex, as shown in Step 2 for SYSA and SYSB.
- ▶ Set the state of each resource element (ON, OFF, or Reset) independently on each system.
- ▶ Define a scheduling environment (which is a set of resource elements, each element having a specified required state) into a single named entity to be used to make scheduling decisions.

The scheduling environment is either *available* (meaning that all the resource elements are in the required state) or *unavailable* (meaning that one or more resource elements are not in the required state).

- ▶ Provide information about the state of the resource elements and the availability of the scheduling environments to interested parties (such as JES or Automation) that make decisions on whether or not the required execution environment is available on an z/OS instance in the sysplex.

Resource element

To understand the idea of resource affinity scheduling, other concepts are introduced, such as:

- ▶ A *resource element* (or scheduling element) is a representation of an execution time resource, which can be ON or OFF in a system. The installation must identify and name these entities.

A resource element can be:

- A resource, such as a database, peripheral device, machine feature, system of cheap cycles
- Or, a TA time property such as second shift or weekend

In Figure 4-2 on page 131, the resource element DB2A is ON in SYSA and OFF in SYSB. This state is set by the installation through the command (F WLM). The resource element PRIMETIME is ON on both z/OS systems.

JCL containing scheduling environment

In Step 3, the JCL of the arriving JP job declares the name of the scheduling environment required (in this case, DB2LATE).

In Step 4, the JP job can only run in the SYSB system. In the SYSB system, the resource element DB2A is OFF and PRIMETIME is ON.

In a multisystem configuration, one challenge is that not every z/OS system is capable of providing all scheduling environments at all times for arriving batch jobs. In other words, the images might not have the same resources. However, it is also important that this be transparent to users.

4.3 Dynamic workload balancing

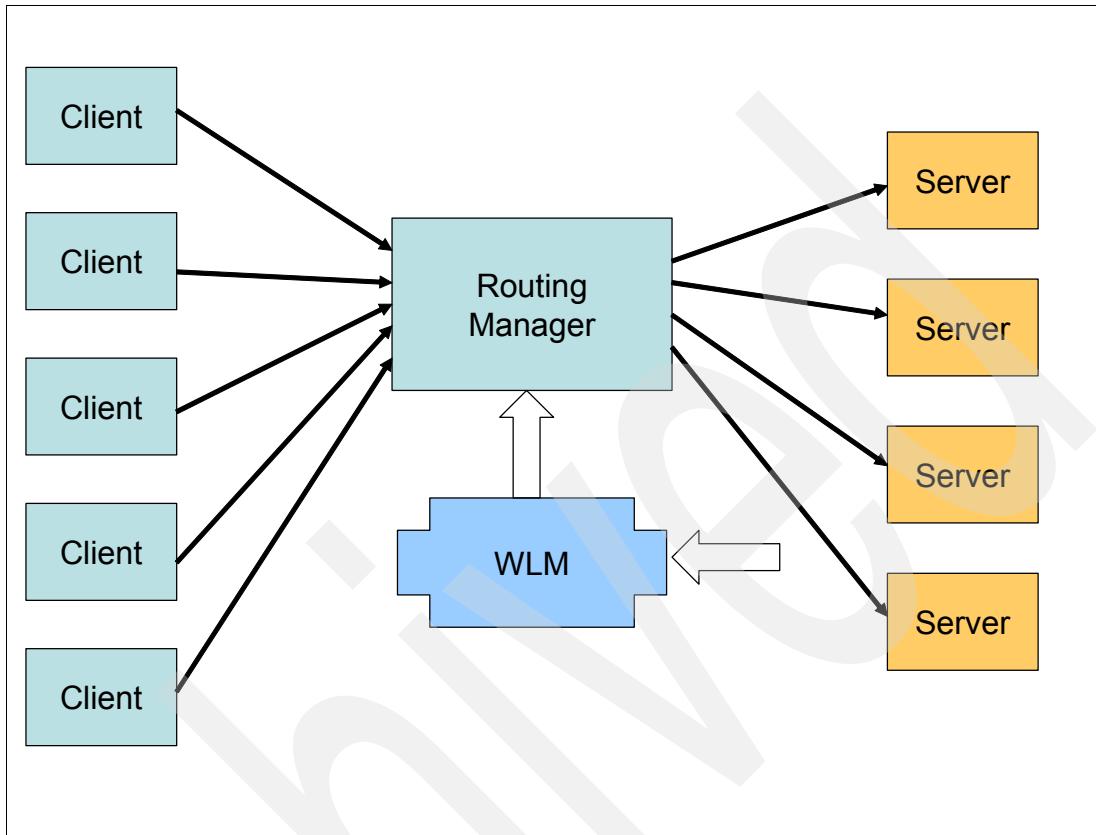


Figure 4-3 Dynamic workload balancing

Dynamic workload balancing

This topic describes how to improve overall sysplex performance when the z/OS systems are cloned (that is, they access the same data and the same program libraries). Based on queuing theory, if you have balanced systems utilization, you may decrease the average transaction queue time.

Dynamic workload balancing implies selecting the best z/OS system and the best address space for an arriving transaction. WLM, because of its global understanding of systems and resources usage, is the right z/OS component to be the advisor for a transaction manager just receiving an incoming transaction. To use WLM as a dynamic workload balancer advisor, the transaction manager must register itself as an exploiter through the WLM API IWMSRSRG.

To make the most of workload management, work needs to be properly distributed so that MVS can manage the resources. It is essential that the subsystems distributing work are configured properly for workload distribution in a sysplex. You do this with the controls provided by each subsystem. For example, in a JES2 and JES3 environment, you spread initiator address spaces across each system.

Transaction managers

Initial cooperation between MVS and the transaction managers (CICS, IMS, DB2) allows you to define performance goals for all types of MVS-managed work. Workload management dynamically matches resources (access to the processor and storage) to work to meet the goals.

CICS, however, goes further by using CICSplex Systems Manager (CICSplex SM) to dynamically route CICS transactions to meet performance goals. CPSM monitors the performance of CICS resources and systems and presents the resources as though they were part of a single system. This type of cooperation greatly improves CICS transaction workload balancing.

Other subsystems also have automatic and dynamic work balancing in a sysplex. For example, DB2 can spread distributed data facility (DDF) work across a sysplex automatically. DB2 can also distribute work in a sysplex through its sysplex query parallelism function. CICS, TSO, and APPC cooperate with VTAM and workload management in a sysplex to balance the placement of sessions. SOM objects can automatically spread servers across a sysplex to meet performance goals and to balance the work.

Archived

4.4 WLM load balancing algorithms

- ❑ WLM dynamic workload balance algorithms:
 - Capacity-based
 - Round robin
 - Goal achievement
 - Balanced utilization
 - Homogeneous servers

Figure 4-4 WLM load balancing algorithms

Workload balancing

In a sense workload balance does not imply that either the transactions or utilizations are evenly distributed. There are several models (algorithms), listed in Figure 4-4, that may be used by WLM to perform workload balancing:

- ▶ Capacity-based attempts to direct transactions to whichever system has the most available capacity (in terms of MSUs). It does not result in balanced utilization if the CPCs are of significantly different speeds. This model is the one most used by WLM workload balance functions.
- ▶ Round robin attempts to route an equal number of transactions to each server. It also does not result in balanced utilization. If the CPCs are of significantly different speeds, then a similar amount of work results in lower utilization on the larger CPC (for example, the SAP PU rotating IO requests through channels).
- ▶ Goal achievement attempts to route to the server that is currently delivering performance that meets or exceeds the installation's objectives. Faster CPCs deliver better response times at higher utilizations than slower CPCs, again resulting in uneven utilization across the CPCs. XCF uses this model for selecting CF links.
- ▶ Balanced Utilization attempts to direct transactions to whichever system has the least utilized processors. z/OS does this with the logical CPUs.
- ▶ Homogeneous servers attempts to send to the same CICS AOR address space any transactions from the same service class.

4.5 Dynamic workload routing support

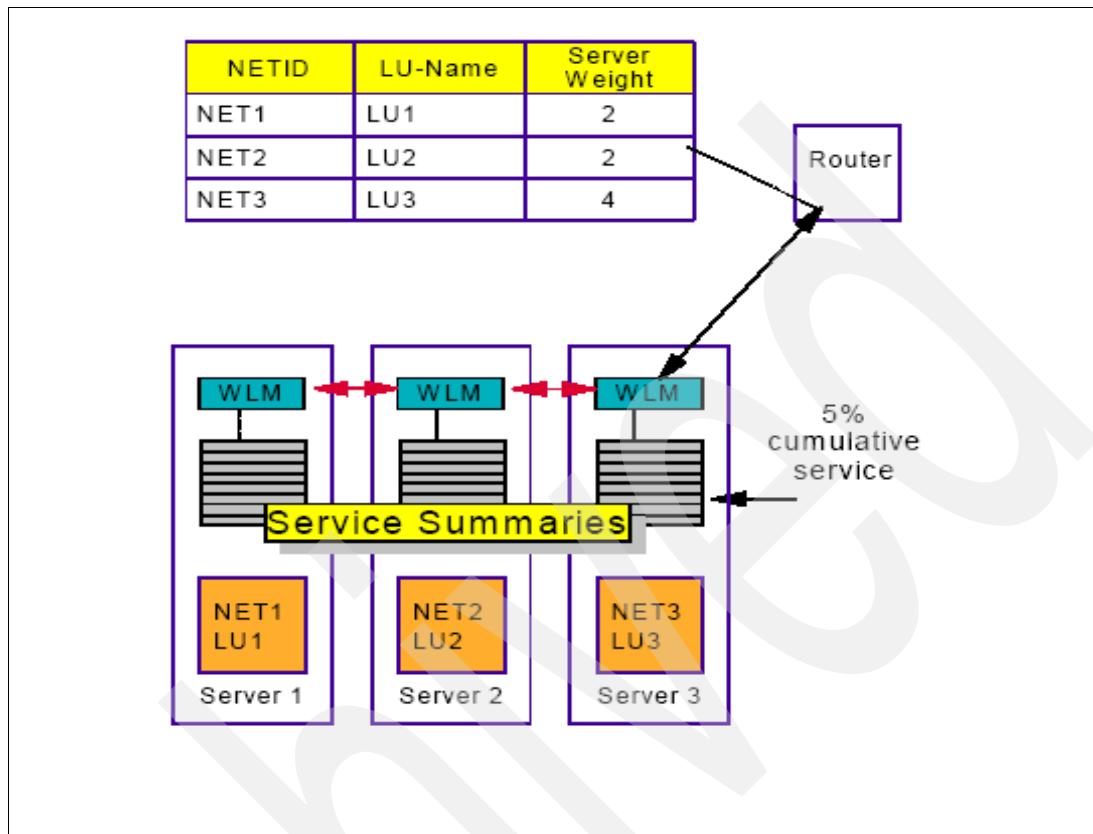


Figure 4-5 Sysplex workload distribution

Sysplex workload distribution

WLM may be called by a transaction manager, for example by the TCP/IP domain name server (DNS), for advice about where (to which server) to route an arriving transaction in a sysplex. This function is invoked by the WLM API WMSRSRS, which provides the caller with a list of registered exploiters with a weight assigned to each one.

An exploiter in this context is a WLM application running in one z/OS, as one CICS AOR. The transaction is then routed by the transaction manager (TCP/IP DNS in the example) to the exploiter proportionally to its weight, so higher-weight z/OS systems receive more transactions.

As shown in Figure 4-5, three application exploiters running in three different z/OS systems (Server 1, Server 2, and Server 3) are returned by WLM with weights of 2, 2, and 4 respectively. DNS then routes one-fourth of the requests to Server 1; one-fourth to Server 2; and one-half to Server 3.

Weight calculation

The weight calculation is as follows:

- ▶ The WLM workload balancer routine measures LP available capacity by:
 - The difference between the actual utilization of this LP and the capacity it is guaranteed as a result of its weight and its number of online logical CPs.

- Plus some portion of the total unused capacity on the CEC, apportioned based on the relative weight of this LP compared to the other LPs on the CEC.
- In some cases, WLM also takes into account the importance of the new transaction, and adds capacity used by lower importance workloads in the available capacity value (because that lower importance work could be displaced to cater for the new, higher importance work).
- ▶ WLM first assigns a weight to each system, with a higher weight to systems with the most available capacity. The weights are proportional to the amount of available capacity on each system. Currently, the weights include zAAP and zIIP available capacity.
- ▶ WLM then divides the weight for a system equally among the registered exploiters on that system. In a general case, you may have multiple exploiters registered with WLM per z/OS. For sysplex distributor, however, the WLM server registration is done by the TCP/IP stack, so there is only one server per system image. This means the server weight always equals the system weight for sysplex distributor. If all systems are running at or near 100% utilization, then higher weights are given to the systems running less important transaction.

WLM selects servers

The objective here is not only to send a transaction where there is available capacity or where the least important transaction was sent (if there is no available capacity); 4.4, “WLM load balancing algorithms” on page 135 explains the algorithms that may be used by WLM to pick a server to route an arrival transaction.

Servers on systems that are in a serious storage shortage (SQA pages, high number of fixed frame, small number of free auxiliary storage slots) do not receive high weights. They are not recommended unless all systems are in a shortage.

Also taken into consideration is the health check of the server (that is, the number of abends per time), to avoid the “black hole effect” (meaning that, because the server is abending, the apparent response time is very short and then even more transactions are sent to the failing server).

4.6 WLM dynamic load balancing exploiters

- VTAM generic resource
- TCP/IP sysplex distributor
- DB2 Connect Enterprise Edition on DDF
- MQSeries load balancing
- CICS CP/SM
- WebSphere Application Server

Figure 4-6 WLM dynamic load balancer exploiters

WLM dynamic load balancing exploiters

Figure 4-6 lists the WLM dynamic load balancing exploiters. They are explained in the following sections in detail.

VTAM generic resource

A *generic resource* is a method of allowing multiple application programs to be known by a common name. The generic resources function allows the assignment of a generic resource name to a group of active application programs that all provide the same function. The generic resource name is assigned to multiple active application programs simultaneously, and VTAM automatically distributes sessions among these application programs rather than assigning all sessions to a single resource. Thus, session workloads in the network are balanced. Session distribution is transparent to users; an LU initiates a logon request using the generic resource name and does not need to be aware of which particular application program is actually providing the function.

The generic resources function also increases application program availability, because each active application program that uses a given generic resource name (a generic resource member) can back up other generic resource members. Thus, no single application program is critical to resource availability. When a generic resource member fails, an LU can reinitiate its session using the same generic resource name. VTAM resolves the session initiation to one of the other generic resource members. Because the user is unaware of which generic resource member is providing the function, the user is less affected by the failure of any single generic resource member.

Sysplex distributor

An example of an internal load balancing solution is the sysplex distributor. Sysplex distributor extends the notion of dynamic VIPA and automatic VIPA takeover to allow for load distribution among target servers within the sysplex. It extends the capabilities of dynamic VIPAs to enable distribution of incoming TCP connections to ensure high availability of a particular service within the sysplex. Sysplex distributor supports load balancing to non-z/OS targets.

Sysplex distributor makes use of Workload Manager (WLM) and its ability to gauge server load and provide a WLM recommendation. In this paradigm, WLM provides the distributing stack with a WLM recommendation for each target system (a WLM system weight), or the target stacks provide the distributing stack with a WLM recommendation for each target server (a WLM server-specific weight). The distributing stack uses this information to optimally distribute incoming connection requests between a set of available servers.

DB2 and DDF

The definition of a response time for the enclaves used to manage DDF transactions depends upon several parameters, including DB2 install parameters and the attributes used when binding the package or plan. DB2 Connect™ Enterprise Edition on DDF connects LAN-based systems and their desktop applications to your company's mainframe and minicomputer host databases.

MQSeries load balancing

MQSeries for z/OS brings the message queue interface (MQI) to your applications. This interface allows you to modify existing applications and to write new applications. The MQI removes much of the need to understand any network or communication systems that you use. Thus, you can expect to complete applications more speedily than before. However, you must plan to take advantage of the MQI by planning its use in your applications, and by understanding the ways in which it assists you.

CICS CP/SM

CICS CP/SM selects the best AOR address space in the best system. WLM controls which requesting (TOR) CICS regions receive the work. WLM can also affect which AOR is chosen when using CICSPlex SM.

WebSphere Application Server

A WebSphere MQ server provides direct connection between service integration messaging engines in WebSphere Application Server and queue managers or queue sharing groups in WebSphere MQ for z/OS. WebSphere MQ server is designed to exploit the high availability and optimum load balancing characteristics provided by a WebSphere MQ for z/OS network. WebSphere MQ server defines the connection and quality of service properties used for the connection, and also ensures that messages are converted between the formats used by WebSphere Application Server, and those used by WebSphere MQ.

4.7 WLM server address space management

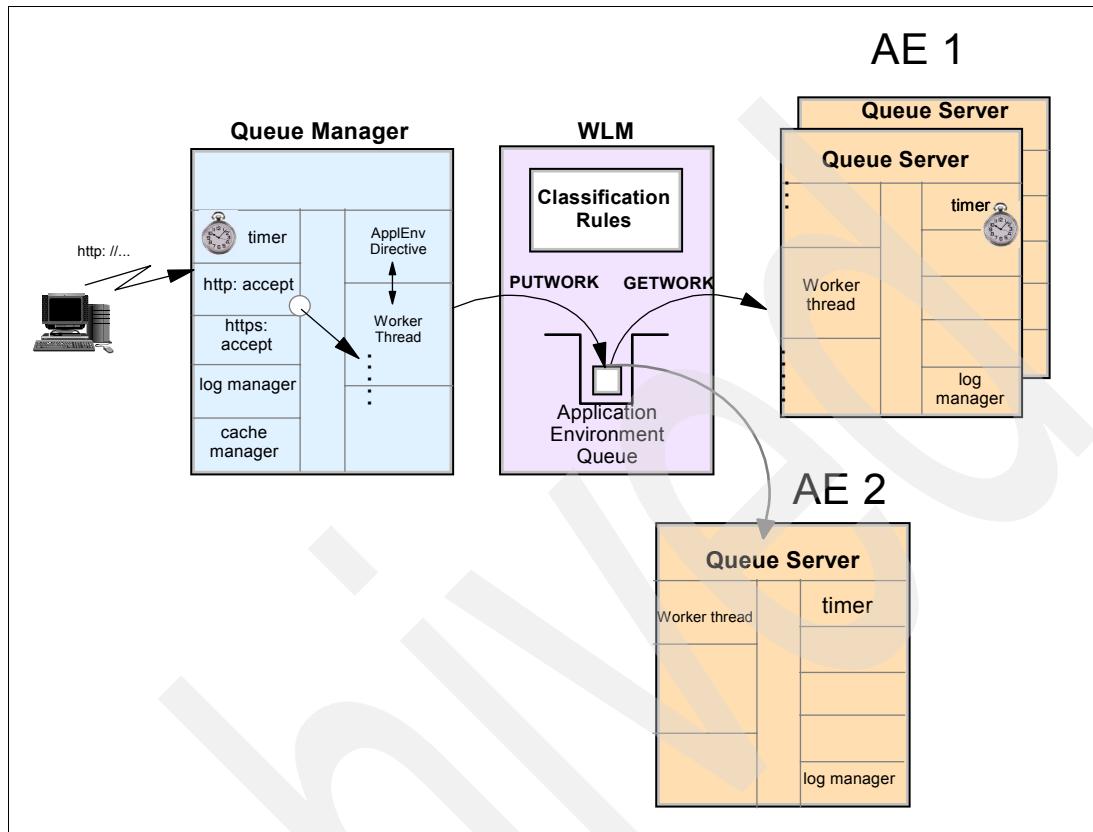


Figure 4-7 Server address space management

Server address space management

Using an application environment is a way to group similar server programs together and have workload management dynamically create and delete server address spaces as needed to handle the work. Each application environment typically represents a named group of server functions that require access to the same application libraries. Depending on the subsystem's implementation of application environments, the scope of server address space management is either confined to a single system or is sysplex-wide. There is also an option to manually start and stop the server address spaces for an application environment if a customer has a special or temporary requirement to control the number of servers independently of workload management. Figure 4-7 illustrates server address space management.

The work manager subsystem contains a table or file that associates the work request names with an application environment name. The application environment names AE1 and AE2 are specified to workload management in the service definition stored in the WLM couple data set, and are included in the active policy when a policy is activated.

Application environment

An *application environment* (AE) is a group of application functions requested by a client that execute in server address spaces. As mentioned, workload management can dynamically manage the number of server address spaces to meet the performance goals of the work making the requests.

Transaction managers

Many transaction managers implement the design of several server address spaces to process in parallel the transaction workload programs.

The design of server address space management poses the following questions:

- ▶ If the goals are not met, can an additional server address space improve the performance index?
- ▶ If the goals are not met, can an additional dispatchable unit in the server address space improve the performance index?
- ▶ If there is a resource constraint (CPU or storage) in the system, how do you reduce the activity of server address spaces?
- ▶ When should the number of server address spaces be decreased?
- ▶ Will the creation of a new server address space adversely impact the performance of other, more important goals?

These transaction managers use WLM services to permit transaction flow from their network-attached address spaces, through WLM, and into server address spaces for execution. The network-attached address spaces are also known as *queue managers* or *controller*, and the server address spaces are also known as *queue servers*, as listed in Figure 4-7 on page 140.

WLM queuing manager

To make this function even more efficient, WLM offers to transaction managers the use of WLM queuing manager services, where the arrival transaction queues (transactions waiting for server address space availability) are managed by WLM. Then WLM is aware of the queue time and when the queue time is affecting the transaction goal. Examples of IBM-supplied transaction managers that use these services are:

- ▶ DB2 for stored procedures
- ▶ Component Broker for WebSphere (CB subsystem in classification rules)
- ▶ HTTP scalable Web server (IWEB subsystem in classification rules)

To implement server address space management, certain transaction managers introduce the concept of application environment; see 4.8, “Application environment” on page 142 for more information about this topic.

JES2 and JES3 transaction managers do not need special application environment definitions to exploit the server address space management functions as implemented in WLM Batch Initiator function; see Figure 4-11 on page 146 for more information about this topic.

4.8 Application environment

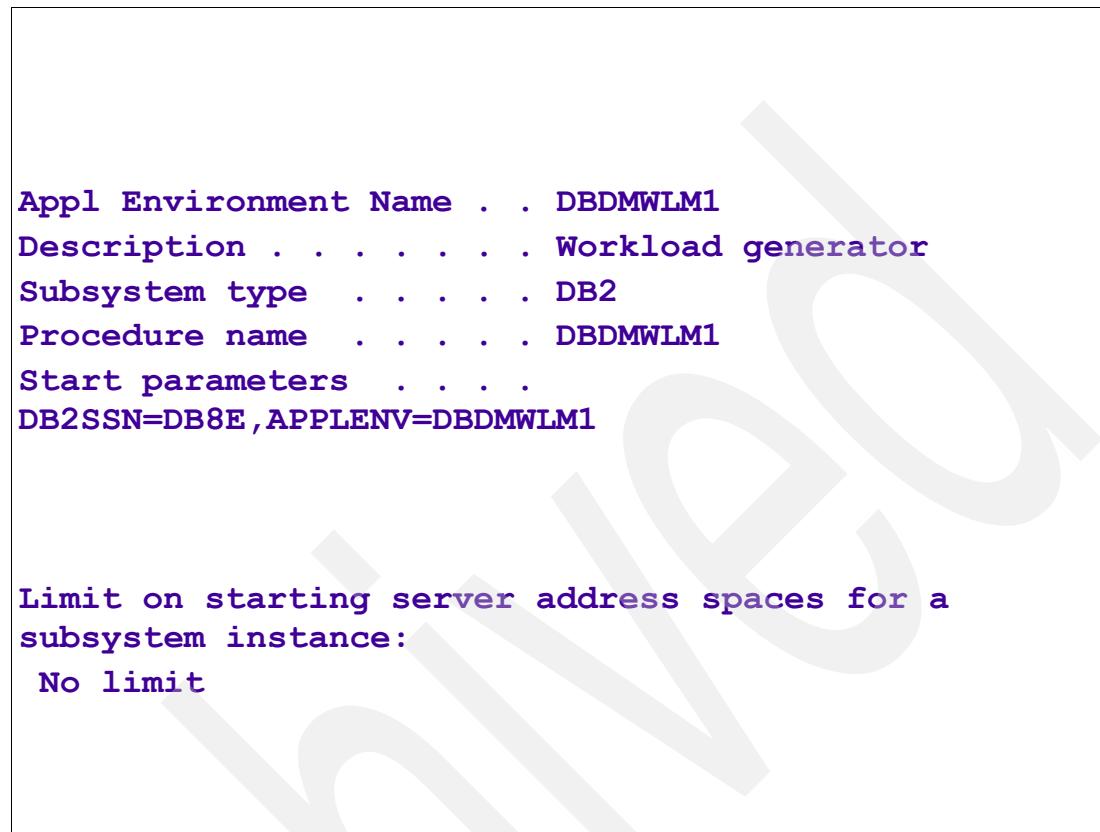


Figure 4-8 Application environment

Application environment

The server address management function is implemented through the application environment concept. An application environment (AE) is a WLM construct grouping server address spaces belonging to a transaction manager that have similar data set definition and security requirements and therefore started by the same JCL procedure. An AE can have a system-wide or sysplex-wide scope, depending of the structure and capabilities of the transaction manager using the AE. The scope of the AE is dictated by the WLM services the transaction manager uses. The queuing manager services provide system scope, and routing services provide sysplex scope for an AE.

Each arriving transaction must have an AE name where it belongs and it only can be selected by a server address space that belongs to this AE name. Then AE is conceived to solve the affinity problem between transactions and server address spaces. If you do not have affinities between transactions and server address spaces, you do not need AEs, or just one AE would be enough.

To summarize, an application environment is used to assign transactions to the correct server address space and also used by WLM to automatically control the number of server address spaces in one AE for transaction managers. Those transaction managers can be DB2 Stored Procedures, SOM, WebSphere Application Server, and MQ, and they all exploit the queuing and routing manager WLM services.

Defining application environments

Application environments are defined in the WLM service definition, as shown in Figure 4-8 on page 142, and allow WLM to do the following:

- ▶ To help meet the goals of a set of transactions by dynamically changing the number of server address spaces
- ▶ And to achieve this without harming other, more important workloads on a system

You must define your AE in the WLM policy and assign incoming transactions to a specific AE. Figure 4-8 on page 142 shows the definition of an AE named DBDMWLM1. The transaction manager specialists need to be informed of that name, then some of its arriving transactions present that name. You may have several AEs per transaction manager.

When defining an AE in WLM service definition, you can specify the following, as shown in Figure 4-8 on page 142:

- ▶ Application environment name - unique in the sysplex
- ▶ Transaction manager - identifies the transaction manager by type and subsystem name
- ▶ STC procedure name - the JCL in SYS1.PROCLIB to start the server address space
- ▶ Start parameters - optional parameters to be passed to the server during address space creation (such as user ID for security checking)
- ▶ WLM management options - special options governing WLM control of address spaces, as NOLIMIT for the number of server address spaces

The workload requirements for a given application may determine that multiple server address spaces should be activated. WLM decides to activate a new address space based on the following information:

- ▶ There is available capacity (CPU and storage) in the system, and transactions in a service class are being delayed waiting on the WLM queue.
- ▶ A service class is missing its goals, it is suffering significant queue delay, and other transactions (donors) in the system can afford to give up resources to the transaction missing goals (receiver).

Application environment queues

Various transactions for a given application environment may have vastly different characteristics, resource consumption patterns, and requirements. For better control by WLM in goal mode, each AE queue is logically divided into a set of sub-queues. Each unique combination of AE and service class defines a single application environment queue, and each server address space can process requests from only one queue.

Note: For every AE queue, WLM starts at least one address space. Therefore, do not specify too many service classes, to avoid having WLM start too many address spaces.

4.9 Dynamic application environment

- ❑ Dynamic application environment allows a transaction manager to dynamically deploy an AE without the need for an installation to explicitly edit the AE parameters in a WLM policy application
 - Middleware is able to define the application environments they need via an API
 - Customer use of the WLM administrative application is not required to define
 - There are no duplicate definitions in middleware and WLM

Figure 4-9 Dynamic application environment

Dynamic application environment

The IWM4AEDF service defines dynamic application environments to WLM. The service can be used by queue manager address spaces to add new application environments after they connect to WLM, and to delete the dynamic application environments before they disconnect from WLM. Figure 4-9 lists dynamic application environment information.

Furthermore, the service can be used to define how server spaces should be resumed for static and dynamic application environments.

Before using this service, the caller must connect to WLM using the IWM4CON service, specifying Work_Manager=Yes, and Queue_Manager=Yes.

A queuing manager must not insert requests for a dynamic and static application environment with the same application environment name concurrently.

4.10 WebSphere and WLM

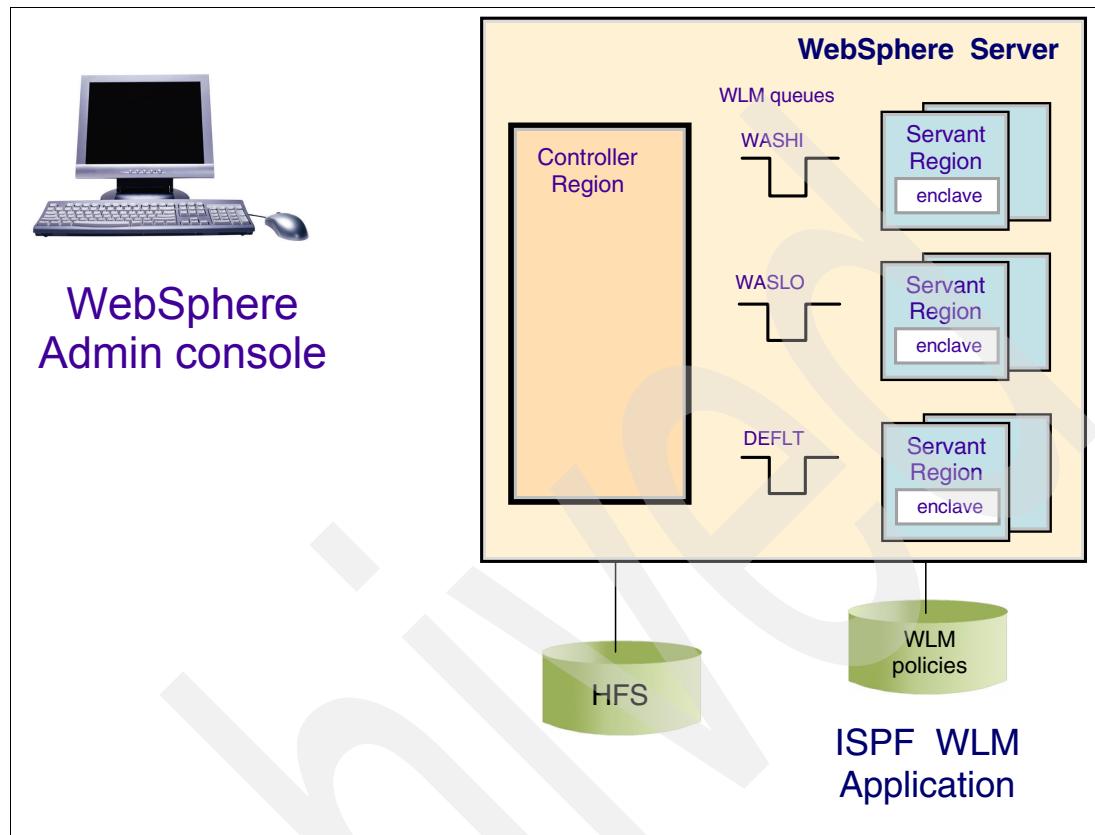


Figure 4-10 WebSphere and WLM

WebSphere and WLM

WebSphere capitalizes on WLM server address space management by using dynamic application environments. WLM classifies a WebSphere Application Server transaction into service classes based on where it arrives, as explained here.

- ▶ If it arrives from the HTTP Server within the same z/OS, it is already associated with an enclave and a service class obtained from the IWEB subsystem.
- ▶ If it arrives from a remote HTTP Server, then an enclave is created and the service class is selected from the CB subsystem in the classification rules.

Server regions

WebSphere applications that run in the servant region as part of the dispatched enclave use WLM classification rules. Each WebSphere transaction is dispatched as a WLM enclave and is managed within the servant region according to the service class assigned through the CB service classification rules.

Server regions process transactions (under TCBs) queued by WLM as follows:

- ▶ The number of tasks per server region depends on server configuration.
- ▶ WLM starts and stops address spaces as needed (and allowed) using the dynamic application environment concept.

4.11 WLM-managed initiators (1)

- ❑ Objective: Manage the time that jobs spend waiting for initiators by controlling dynamically the number of initiators
- ❑ Two types of initiators: WLM managed and JES2/JES3 managed
- ❑ Implementation:
 - Specify JOBCLASS(x), MODE=WLM in JESPARM for JES2
 - Specify GROUP,NAME=x,MODE=WLM for JES3
 - Jobs of the same WLM service class should not have classes from both JES2/JES3 and WLM initiators
 - Adjust service class goals for queue delays

Figure 4-11 WLM batch initiators

WLM-managed initiators

JES2 and JES3 provide automatic and dynamic placement of initiators for WLM-managed job classes. JES2 and JES3 provide initiator balancing, so that already available WLM-managed initiators can be reduced on fully loaded systems and increased on low loaded systems to improve overall batch work performance and throughput over the sysplex. With WLM initiator management, initiators are managed by WLM according to the service classes and performance goals specified in the WLM policy.

The main purpose of WLM batch initiator management is to let WLM dynamically adjust the number of active initiators. In this way, WLM can manage the time that jobs wait for an initiator based on the goals of the batch service class periods.

JES2 and JES3 WLM batch initiators

This example describes WLM batch initiator management in JES2 and JES3 environments. Some of these are external changes and they imply an installation action:

- ▶ Define WLM-managed job classes, with the number of initiators limited by a maximum value, via JES2 job class parameters in JES2 parms (or a \$T **jobclass** command):

```
JOBCLASS=Q, MODE=WLM, XEQCOUNT=MAX=nnnn
```

In this example, jobs from class Q are selected by a WLM-controlled initiator associated with the job's service class. JES2 classifies (asks WLM for a service class) the batch job

earlier, at the end of job conversion rather than at job selection. This allows WLM to manage the job's queue delay time according to the service class goal.

As opposed to an application environment, here the queue is managed by JES2 and WLM is informed when the JOB enters and when it leaves the queue. With this information, WLM can track the delay for initiator figures.

- ▶ In this JES3 example, there are three systems in the SYSPLEX and you want group JES3TEST to run only on two of those systems. Because the initiator options are not specified, the default options will be used. The EXRESC parameter is specified as follows:
`GROUP,NAME=JES3TEST,MODE=WLM,EXRESC=SY1,EXRESC=SY2`

JES initiator management

JES2 and JES3 maintain two different queue organizations for all jobs awaiting execution (initiator selection), as explained here:

- ▶ All jobs are queued by job class, priority, and the order in which they finish conversion. This is the queue from which JES2-managed initiators select jobs for execution.
- ▶ Jobs awaiting execution in WLM-managed job classes are also queued by their WLM-assigned service class in the order they were made available for execution.
- ▶ WLM relies upon additional transaction queuing delay information provided by JES2. In the case of application environment, WLM (as in WebSphere transactions) does not need that because the transaction queue is managed by WLM.
- ▶ The policy adjustment routine triggers a START INIT command when the major delay for a batch service class period is the delay-for-initiator. This starting initiator first serves the service class period suffering these delays.

The decision to increase the number of WLM initiators takes into consideration the effect on general system performance, and sometimes such an increase is not executed.

4.12 WLM-managed initiators (2)

- Individual JES2/JES3 batch queues can be WLM-managed
- Initiators are dynamically started by WLM to meet service class period goals by reduction of batch execution delays
- WLM selects the system based on:
 - Available system resources
 - Existence of initiator waiting batch jobs
 - Importance of the service class
- Initiators are dynamically stopped by WLM:
 - When significantly more initiators exist than needed (> 1.5 times avg. queue length)
 - In case of CPU or memory shortage
 - After one hour of inactivity

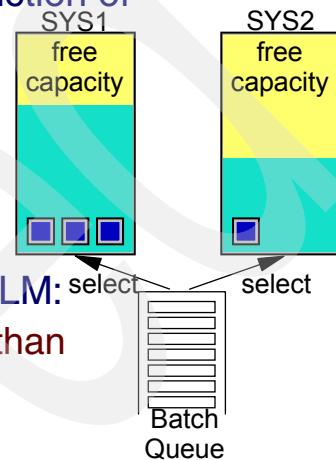


Figure 4-12 WLM-managed initiators

WLM batch initiators

WLM initiator balancing manages WLM-managed batch initiators so that job starts are preferred on low utilized systems, rather than on high utilized ones, as long as affinity requirements do not restrict jobs to dedicated systems. Figure 4-12 lists the WLM batch initiators.

WLM improves the balancing of WLM managed batch initiators between systems of a sysplex. Although in earlier releases a balancing of initiators between high and low loaded systems was only done when new initiators were started, this is now done when initiators are already available. On highly utilized systems, the number of initiators is reduced while new ones are started on low utilized systems.

This enhancement can improve sysplex performance with better use of the processing capability of each system. WLM attempts to distribute the initiators across all members in the sysplex to reduce batch work on highly used systems, while taking care that jobs with affinities to specific systems are not hurt by WLM decisions.

Initiators are stopped on systems that are utilized over 95% when another system in the sysplex offers the required capacity for such an initiator. WLM also increases the number of initiators more aggressively when a system is low utilized and jobs are waiting for execution.

WLM job selection

Jobs are selected for execution within a service class by their main service arrival time. For the most part, this is the time that the job completed C/I processing. Priority does not influence whether a job is selected for execution first, so changing the priority will have no effect, unless it causes the job's service class to be changed.

4.13 z/OS UNIX performance overview

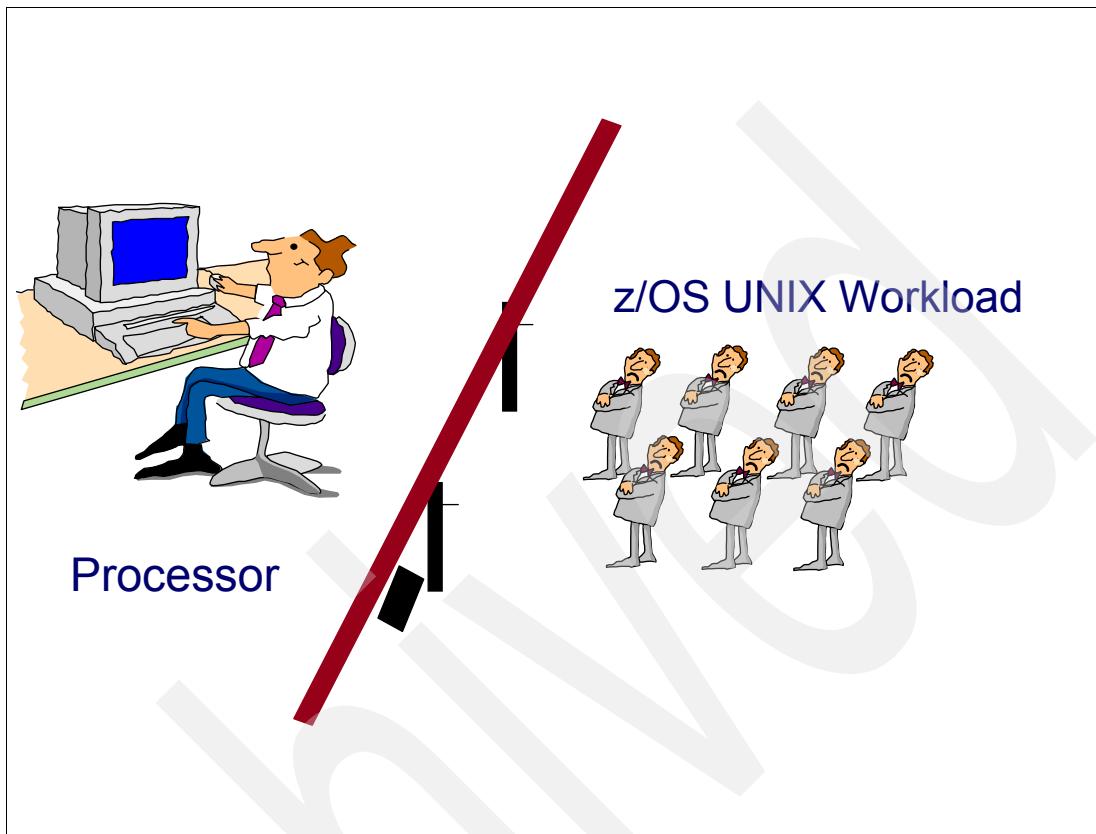


Figure 4-13 A z/OS UNIX performance overview

z/OS UNIX performance overview

It is both important and necessary to tune the z/OS UNIX environment to obtain an acceptable level of performance; see Figure 4-13. Because z/OS UNIX is tightly integrated into the operating system, there are many z/OS UNIX tuning steps necessary to reach maximum interoperability between z/OS UNIX and traditional z/OS services.

As you run more UNIX-based products, you need a z/OS UNIX environment that performs well. Such products include the Lotus® Domino Server, TCP/IP, SAP R/3, and Novell Network Services based on z/OS UNIX. Keep the following considerations in mind regarding your z/OS UNIX workload:

- ▶ Each z/OS UNIX interactive user consumes up to double the system resource of a TSO/E user.
- ▶ Every time a user tries to invoke the z/OS UNIX shell, RACF will have to deliver the security information out of the RACF database.
- ▶ The average active user will require three or more concurrently running processes that, without tuning, run in three or more concurrently active address spaces.

These are only a few of the considerations regarding performance impacts and how to prevent them. In most cases in our lab, tuning improved throughput by 2 to 3 times and the response time improved by 2 to 5 times. The following sections describe how to achieve such a significant improvement.

4.14 WLM in goal mode

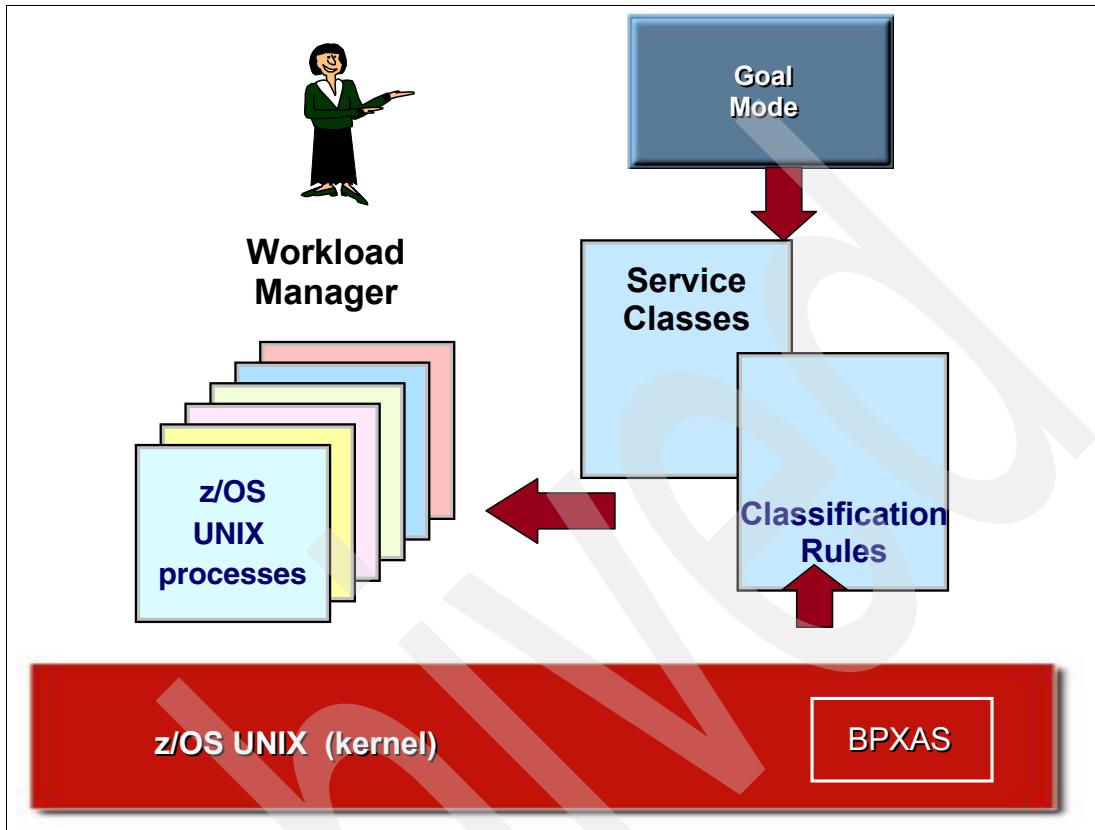


Figure 4-14 Using WLM in goal mode

Goal mode definitions for z/OS UNIX processes and daemons

Installations that run in goal mode, illustrated in Figure 4-14, can take the following steps to customize service policies in their Workload Manager service definition:

1. Define a workload for z/OS UNIX kernel work.
2. Define service classes for z/OS UNIX kernel work:
 - a. Define a service class for forked children.
 - b. Define a service class for startup processes.
3. Define classification rules:
 - a. By default, put child processes (subsystem type OMVS) into the service class defined for forked children.
 - b. Put the kernel (TRXNAME=OMVS) into a high-priority started task (subsystem type STC) service class.
 - c. Put the initialization process BPXOINIT (TRXNAME=BPXOINIT) into a high-priority started task (subsystem type STC) service class.
 - d. Startup processes that are forked by the initialization process, BPXOINIT, fall under SUBSYS=OMVS.
 - e. Other forked child processes (under subsystem type OMVS) can be assigned to different service classes.
 - f. Put the DFSMS buffer manager SYSBMAS (TRXNAME=SYSBMAS) into a high-priority started task (subsystem type STC) service class.

4.15 Capping and WLM

- ❑ Capping is used to limit, artificially, the CPU consumption rate of a specific set of dispatchable units, usually associated with user transactions
- ❑ Types of capping in zSeries, z9 and z10:
 - WLM resource group capping, where just a set of service classes are capped
 - LPAR capping (through weights or by decreasing the number of logical CPs), where the full LPAR is capped
 - Soft capping (LPAR plus WLM), where the full LPAR is capped
 - Discretionary capping, where overachieving service classes are capped

Figure 4-15 *Types of capping*

Capping and resource groups

If work in a resource group is consuming more resources than the specified maximum capacity, the system caps the associated work accordingly to slow down the rate of resource consumption. The system may use several mechanisms to slow down the rate of resource consumption, including swapping the address spaces, changing its dispatching priority, and capping the amount of service that can be consumed. Reporting information reflects that the service class may not be achieving its goals because of the resource group capping.

By setting a minimum processing capacity, you create an overriding mechanism to circumvent the normal rules of importance. If the work in a resource group is not meeting its goals, then workload management will attempt to provide the defined minimum amount of CPU resource to that resource group.

Types of capping

Using capping artificially limits the CPU used rate for a specific set of tasks and service requests usually associated with user transactions. The types of capping, as listed in Figure 4-15, are:

- ▶ With WLM resource group capping, a set of service classes in a sysplex is capped.
- ▶ LPAR capping is achieved through weights or by decreasing the number of logical CPs. The full LPAR is capped.
- ▶ With soft capping by LPAR with WLM, the full LPAR is capped.

- ▶ With WLM discretionary capping, overachieving work is capped to help the discretionary work.

Reasons to use caps

The use of all capping types is optional. The reasons for capping can be:

- ▶ To save software fees in a Workload License Charger (WLC) type of software contract
- ▶ In a service bureau, where a company customer should not consume more than they are paying for
- ▶ To limit I/O access from a less important workload to a common I/O DASD controller
- ▶ To protect your most important transactions from z/OS non-full preemptible behavior
- ▶ To emulate a Quiesce action for a non-swappable address space

4.16 Resource group capping

- ❑ A resource group is a way to limit or guarantee CPU capacity to transactions running in service classes:
 - Assign to one or more service classes
 - Amount of sysplex capacity in CP service unit per second
 - Two values, one limiting or capping (maximum) and other guaranteeing (minimum)
 - Resource groups are optional

Figure 4-16 Resource groups

Resource groups

Using resource groups is a way to limit or guarantee resource capacity. A *resource group* is a named amount of CPU capacity that you can assign to one or more service classes. For most systems, you can let workload management decide how to manage the resources in the sysplex and not use resource groups. You set performance goals for work and let workload management adjust to meet the goals.

In some cases, however, you might want to use a resource group to, for example, limit the service that a certain service class can consume. It is recommended that you assign each resource group to only one service class.

Using resource groups

You can use a resource group to perform the following tasks:

- ▶ Limit the amount of processing capacity available to one or more service classes.
- ▶ Set a minimum processing capacity for one or more service classes if the work is not achieving its goals.
- ▶ Define a minimum and maximum amount of capacity sysplex-wide, or at a system level.
- ▶ Limit capping by an amount of CPU capacity available to some service classes. Capping is used in situations where you want to deny the access of CPU cycles to one or more service classes.

- Guarantee a minimum CPU capacity to the service classes when a transaction in the group is missing its goals.

Defining resource groups

You can specify a minimum and maximum amount of capacity to a resource group. You can assign only one resource group to a service class. You can assign multiple service classes to the same resource group. You can define up to 32 resource groups per service definition.

Keep in mind your service class goals when you assign a service class to a resource group. Given the combination of the goals, the importance level, and the resource capacity, some goals may not be achievable when capacity is restricted.

If work in a resource group is consuming more resources than the specified maximum capacity, then the system caps the associated work accordingly to slow down the rate of resource consumption. The system may use several mechanisms to slow down the rate of resource consumption, including swapping the address spaces, changing its dispatching priority, and capping the amount of service that can be consumed. Reporting information reflects that the service class may not be achieving its goals because of the resource group capping.

Note: Resource group capping, as a goal enforcement, is performed at the sysplex level. The CPU service rate values are accumulated first on local systems, and then across the sysplex for total. The total value of the consumed service rate is used to determine if it exceeds the resource group maximum service rate.

4.17 WLM resource groups

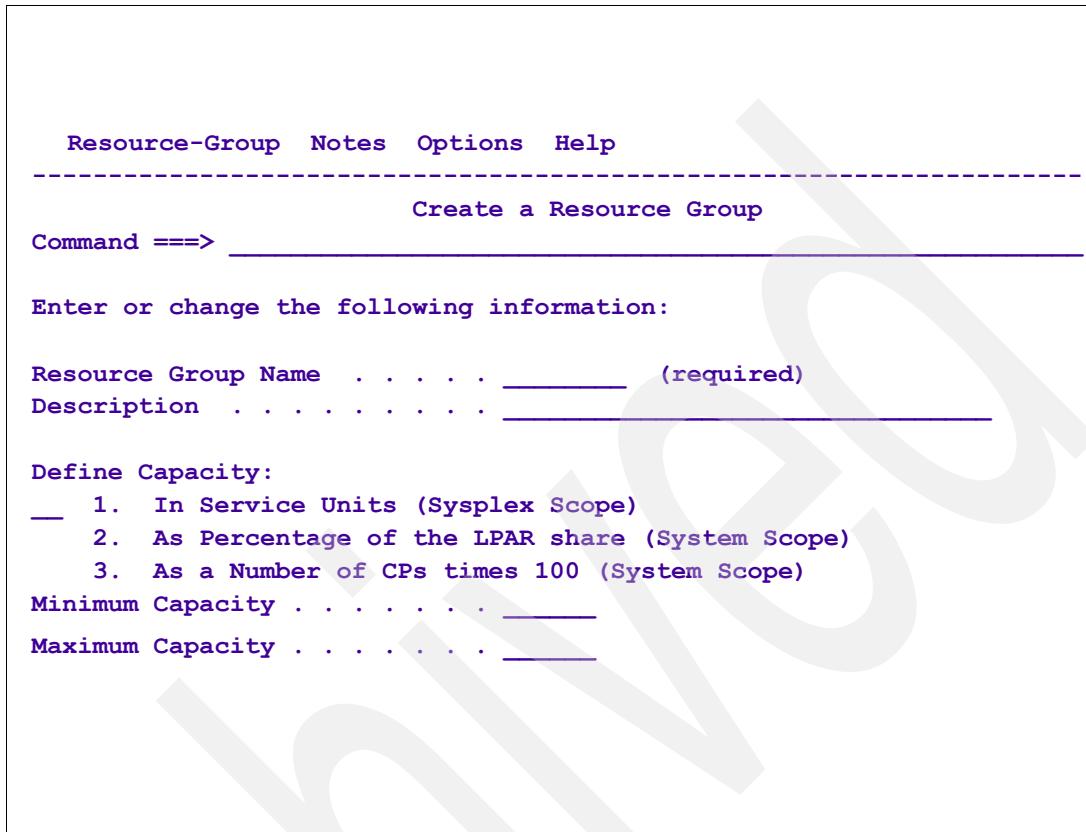


Figure 4-17 Resource group definition panel

Defining a resource group

Figure 4-17 displays a resource group definition panel. There are three resource group types, as listed here.

- ▶ **Resource group type 1**

With resource group type 1, the capacity is specified in unweighted CPU service units per second. The value must be between 0 and 999999. The minimum and maximum capacity applies sysplex-wide; that is, WLM ensures that the limits are met within the sysplex.

- ▶ **Resource group type 2**

With resource group type 2, the capacity is specified as a percentage of the LPAR share. The value must be between 0 and 99, and the sum of all minimum LPAR share percentages for all resource groups of this type should not exceed 99. The minimum and maximum capacity has a system scope; that is, WLM ensures that the limits are met on each system within the sysplex.

- ▶ **Resource group type 3**

With resource group type 3, the capacity is specified as a number of general purpose processors (CPs). A number of 100 represents the capacity of one CP. The number should be between 0 and 999999. The minimum and maximum capacity has a system scope; that is, WLM ensures that the limits are met on each system within the sysplex.

This book focuses on resource group type 2 and 3, as detailed in “Resource group - type 2” on page 158 and “Resource group - type 3” on page 159.

To create a new resource group, enter a name and a description. You can also specify a minimum and maximum amount of capacity in CPU service units across the sysplex, or percentage value within the system, or number of General Purpose Processors (CPs)' for example:

```
Resource Group Name . . . BATCHHI  
Description . . . . . Guaranteed CPU access  
Define Capacity  
 1_ 1. In Service Units (Sysplex Scope)  
    2. As Percentage of the LPAR share (System Scope)  
    3. As a number of CPs times 100 (System Scope)  
Minimum Capacity . . . 5000  
Maximum Capacity . . . 7000
```

Specify a one- to eight-character name of the resource group. Every resource group name must be unique within the sysplex.

When defining a resource group, consider how workloads and service classes have been set up, including whether they have been set up by subsystem, by application, or by department or location. You can define up to 32 resource groups per service definition. For options 2 or 3, the minimum or maximum capacity is defined on a system level, which means that every system is individually managed to ensure the minimum and maximum capacity.

Minimum and maximum capacity

As previously mentioned, you can specify a minimum and maximum amount of capacity to a resource group. You can assign only one resource group to a service class. You can assign multiple service classes to the same resource group. You can define up to 32 resource groups per service definition.

The capacity is specified in unweighted CPU service units per second. The value must be between 0 and 999999.

Minimum and maximum capacity applies sysplex-wide; that is, WLM ensures that the limits are met within the sysplex.

Minimum This refers to the CPU service that should be available for this resource group when work in the group is missing its goals. The default is 0. If a resource group is not meeting its minimum capacity and work in that resource group is missing its goal, then workload management will attempt to give CPU resource to that work, even if the action causes more important work (outside the resource group) to miss its goal.

If there is discretionary work in a resource group that is not meeting its minimum capacity, then WLM will attempt to give the discretionary work more CPU resource if that action does not cause other work to miss its goal.

The minimum capacity setting has no effect when work in a resource group is meeting its goals.

Maximum This refers to the CPU service that this resource group may use. Maximum specified for this resource group applies to all service classes in that resource group combined.

Maximum is enforced. There is no default maximum value.

4.18 Resource group - type 2

Create a Resource Group

Command ==> _____

Enter or change the following information:

Resource Group Name RGLPMX50 (required)
Description Limit to 50% of an LPAR

Define Capacity:

2 1. In Service Units (Sysplex Scope)
2. As Percentage of the LPAR share (System Scope)
3. As a Number of CPs times 100 (System Scope)

Minimum Capacity _____
Maximum Capacity 50_____

"LPAR share" is minimum of:

- CEC CPU pool capacity * LP weight / (sum of weights)
- LP capping capacity
- Defined capacity limit (when soft capping is in effect)

Figure 4-18 Percentage of LP share resource group capping

Percentage of LP share resource group capping

Beginning with z/OS V1R8, there are two additional options (both with a numerical system scope).

Resource group - type 2 - a percentage of the LP share weight

For resource group type 2, the capacity is specified as a percentage of the LPAR share. The value must be between 0 and 99, and the sum of all minimum LPAR share percentages for all resource groups of this type should not exceed 99.

In Figure 4-18, the resource group named RGLPMX50 limits its assigned service classes to no more than 50% of the logical partition (LP) upon which that service class transactions are running. If this constraint comes into play on one LP, it has no effect on the amount of covered transactions running on another LP *unless* that transaction also tries to use more than 50% of that LPAR. Note that the figure pictured here is enforced at z/OS scope, but the number is the same for every z/OS in the sysplex covered by this WLM policy. Therefore, the transactions experience different constraints on different LPARs.

LPAR share

The LPAR share is defined as the percentage of the weight definition for the logical partition to the sum of all weights for all active partitions on the CEC. Note that the definition of "capacity" (LPAR share) depends on other constraints that may have been placed on the LPAR such as number of logical CPUs, soft capping, and LPAR capping.

4.19 Resource group - type 3

Create a Resource Group

Command ==> _____

Enter or change the following information:

Resource Group Name RGCPX250 (required)
Description Limit to 250% of a CP

Define Capacity:
3 1. In Service Units (Sysplex Scope)
2. As Percentage of the LPAR share (System Scope)
3. As a Number of CPs times 100 (System Scope)

Minimum Capacity _____
Maximum Capacity 250 _____

Limits are in % of the LP's CP engine

- Subject to knee-capping
- Subject to IRD
- May reflect different SU/sec limits on different LPs

Figure 4-19 Resource group - type 3

Resource group - type 3

For resource group type 3, the capacity is defined as a number of general purpose processors (CPs). The following example illustrates how capping works for resource group type 3 and explains how to calculate this by determining how many service units per second (SU/s) the defined capacity corresponds to.

Note: An advantage of using resource group type 3 is that it dynamically adjusts to the processor capacity when the work is run on another hardware.

Figure 4-19 displays the panel for the type 3 option showing a percentage of the total CEC capacity in units of CPUs multiplied by 100.

Resource group example

In this example, resource group RGCPX250 limits its assigned service classes to no more than 250% of the CP speed (equivalent to 2.5 CPUs) for the LP upon which those service class transactions are running.

If this constraint comes into play on one LP, it has no effect on the amount of covered transactions running on another LPAR, unless those transactions also try to use more than 2.5 CPUs.

Note that the definition of CP speed is subject to external manipulation, as follows:

- ▶ If the hardware has been “dialed back” to represent a slower machine
- ▶ If IRD has altered the LPAR’s relative weight

Therefore, the workload will experience different constraints on different LPARs.

Resource group types

Summarizing resource group types:

- ▶ Type 1 resource group (SU/sec) is classic, sysplex-wide.
- ▶ With a type 2 resource group (% of an LPAR), each LPAR is treated separately, which simplifies the migration to a new hardware configuration.
- ▶ With a type 3 resource group (% of an engine), each LPAR is treated separately, which simplifies both the definitions and explanations.

4.20 Soft capping

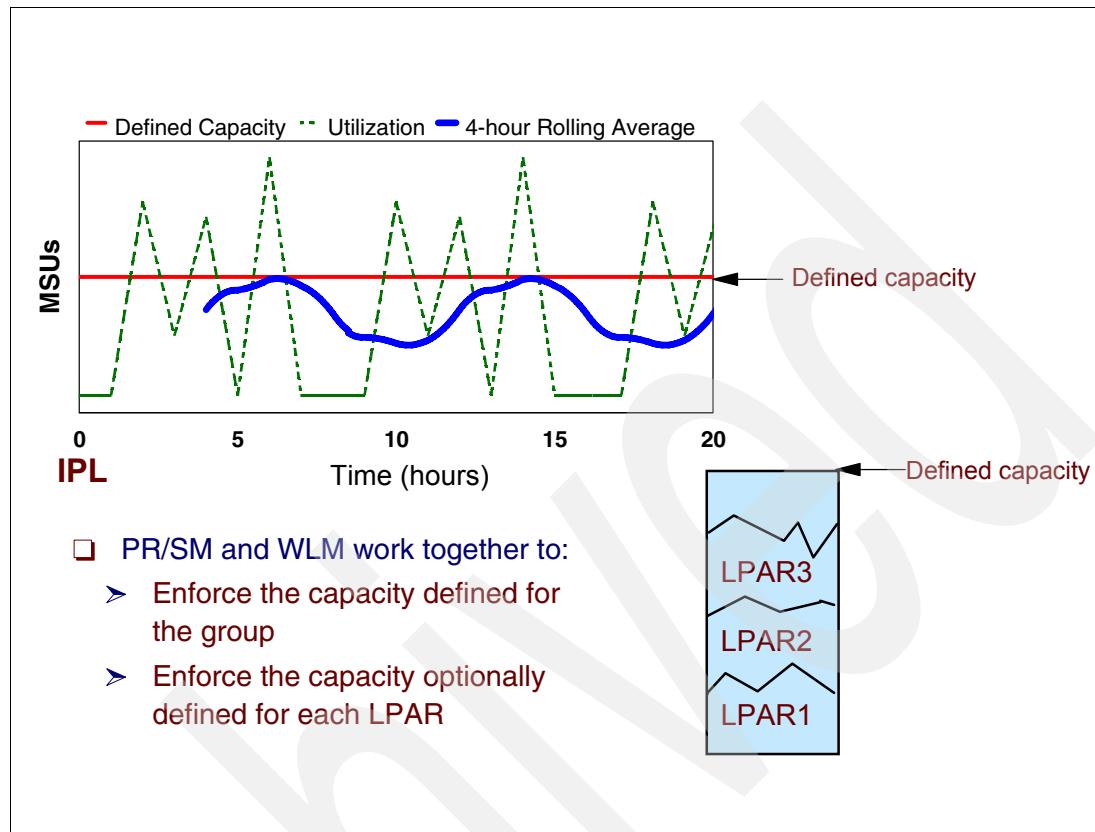


Figure 4-20 Soft capping

Defined capacity

Defined capacity is part of the z/OS support of Workload License Charges. With it, you can set a defined capacity limit, also called a soft cap, for the work running in a logical partition. This defined capacity limit is measured in millions of service units per hour (MSUs). It allows for short-term spikes in the CPU usage, while managing to an overall, long-term, 4-hour rolling average. It applies to all work running in the partition, regardless of the number of individual workloads that the partition may contain.

4-hour rolling average

WLM enforces the defined capacity limit by tracking the partition's CPU usage and continually averaging it over the past 4 hours. Spikes above the defined capacity limit are possible, as shown in Figure 4-20 with the dashed line, as long as they are offset by low points that keep the 4-hour average at or below the defined capacity limit. When this 4-hour average goes over the defined capacity limit, however, then WLM caps the partition (soft cap). At that point, it can use no more than the defined capacity limit, until the average drops below the limit.

Group capacity

Soft capping (also known as "defined capacity") is another type of capping that has been implemented to support usage-based software pricing. In z/OS V1R8 and z/OS V1R9, the WLM defined capacity mechanism is extended to handle LPAR groups instead of a single LPAR. This is called "group capacity limit support." Because the use of defined capacity began some years ago, there is a requirement to have more flexibility when defining capacity

limits for LPARs, so an enhanced mechanism is needed. The group capacity limit balances the capacity between groups of partitions on the same processor. This requirement, related to WLM, requires the use of IBM System z9® and z/OS V1R8 and higher. The software support allows grouping of LPARs in the same processor and the LPARs are then managed on a group basis using the existing WLM defined capacity mechanism.

Group capacity example

In Figure 4-20 on page 161, each partition (LPAR1, LPAR2, and LPAR3) manages itself independently from all other partitions and the group capacity you define is based on defined capacity. Therefore, a 4-hour rolling average of the group MSU consumption is used as a base for managing the partitions of the group.

Each partition is going to see the consumption of all the other LPARs on the processor. If the partition belongs to a group, it identifies the other partitions in the same group and calculates its defined share of the capacity group based on the partition weight (compared to the group).

This share is the target for the partition if all partitions of the group want to use as much CPU resources as possible. If one or more LPARs do not use their share, this donated capacity is distributed over the LPARs that need additional capacity. Even when a partition receives capacity from another partition, it never violates its defined capacity limit (if one exists).

4.21 Group capacity capping

- ❑ WLM uses the weight definitions of the partitions and their actual demand to decide how much CPU can be consumed by each partition in the group.
- ❑ Capacity groups are defined consisting of three partitions: LPAR1, LPAR2, and LPAR3. The group limit is defined to 50 MSU and the weights of the partitions are shown.

Partition	Weight	Share [MSU]
LPAR1	100	16.7
LPAR2	50	8.3
LPAR3	150	25

Figure 4-21 Group capacity capping

Group capacity capping

Group capacity limit is an extension of the defined capacity. It allows an installation to define a soft cap for multiple logical partitions of the same CEC, all running z/OS V1R8 or higher. The group limit is a defined capacity (or soft cap) for all partitions defined in the group. The capacity group is defined on the Hardware Management Console (HMC). Each capacity group has a name and a defined capacity that becomes effective to all partitions in the group.

WLM uses the weight definitions of the partitions and their actual demand to decide how much CPU can be consumed by each partition in the group.

This is an extension in z/OS V1R8 of defined capacity (soft capping), by adding more flexibility. Instead of soft capping each LP individually, the installation caps a group of LPARs containing shared logical CPUs.

A capacity group must be defined using the following rules:

- ▶ It consists of multiple LPARs on the same processor.
- ▶ LPARs must run at least z/OS V1R8 or higher.
- ▶ It is possible to define multiple groups on a processor.
- ▶ A partition can only belong to one group.
- ▶ A group member can also have a defined capacity.
- ▶ A capacity group is independent of a sysplex and an LPAR cluster.

Group capacity example

The total weight of all partitions in the group is 300. Based on the weight definitions, each partition gets an entitled share of the group capacity of 50 MSU. The entitled share is important to decide how much MSU can be used by each partition if the 4-hour rolling average of the group exceeds the group capacity limit. The share is also shown in Figure 4-21 on page 163.

In this example, at 07:00 p.m. when the systems are IPLed, all three partitions are started. In the beginning, only partition LPAR1 and LPAR2 use approximately 60 MSU. No work is running on partition LPAR3. Therefore its measured consumption is very small.

WLM begins to cap the partitions when the 4-hour rolling average for the combined usage exceeds the 50 MSU limit. This happens around 09:00 p.m. At that point, LPAR1 is reduced to about 30 MSU and LPAR2 is reduced to about 20 MSU. LPAR3 still does not demand much CPU. Therefore, the available MSU of the group can be consumed by LPAR1 and LPAR2.

Around 11:00 p.m., work is started on LPAR3. A small spike can be observed when WLM recognizes that the third partition starts to demand its share of the group. After that spike, LPAR3 gets up to 25 MSU of the group because its weight is half of the group weight. MVS1 is reduced to 16.7 MSU and MVS2 is reduced to 8.3 MSU. Based on variation in the workload, the actual consumption of the partitions can vary but the group limit of 50 MSU is always met on average.

4.22 WLM discretionary work capping

- Must not currently be a receiver or donor
- Must not be a member of a resource group
- Must not be a small consumer (I/O-bound)
- Must be the last uncapped non-discretionary period
- Velocity goal is less than or equal to 30%
- Response time goal is more than 1 minute
- Projected to have a PI better than 0.7
- Has not been a significant receiver candidate for CPU

Figure 4-22 Discretionary capping

Discretionary capping

Certain types of work, when overachieving their goals, potentially will have their resources “capped” to give discretionary work a better chance to run. Figure 4-22 lists discretionary capping rules.

Specifically, work that is not part of a resource group and has one of the following types of goals will be eligible for this resource donation:

- ▶ A velocity goal of 30 or less
- ▶ A response time goal of over one minute

Note: Work that is eligible for resource donation is work that has been significantly overachieving its goals. If you have eligible work that must overachieve its goals to provide the required level of service, then adjust the goals to more accurately reflect the work's true requirements.

This function is not about capping dispatchable units running in the discretionary type of goal service class. It is about capping other dispatchable units to allow discretionary work to run at all (that is, to obtain some CPU cycles) in a busy system.

Discretionary goals

Usually a service class period with a discretionary goal can obtain CPU service when all the listed service class periods are in wait or active in other PUs. Other way of getting PU is by WLM capping internally happy service class periods (PIs of less than one) that are over-achieving their goal. This is done to deliver some CPU resource to the discretionary service class periods, but also guarantees that the happy service class periods will remain happy (but less happy). As a consequence, during this capping interval the transactions with a discretionary goal can sometimes get more CPU than the non-discretionary transactions.

This is implemented by capping internally (no external parameters) the happy service class periods that have a dispatch priority higher than the discretionary ones. Figure 4-22 on page 165 lists rules to use to apply such capping. Notice that such rules are very drastic, thus making discretionary an almost rare event.

Note that *all* of the conditions must be met, meaning that WLM does not cap managed transactions arbitrarily.

4.23 WLM z10 EC capacity provisioning

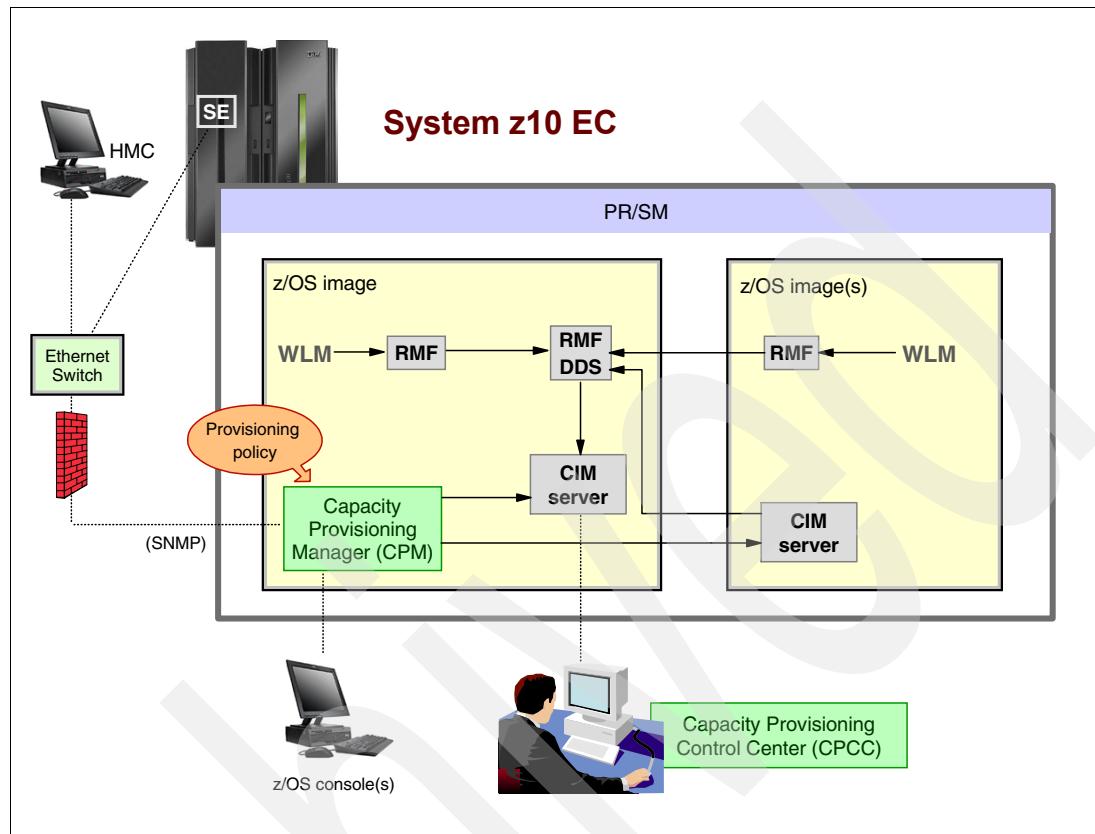


Figure 4-23 Capacity Provisioning overview

Capacity provisioning overview

z/OS Workload Manager (WLM) manages the workload by goals and business importance (as defined by an installation WLM policy) on each z/OS system in a Parallel Sysplex. WLM metrics (as resource delays and performance index) are available through existing interfaces and are reported through RMF Monitor III, with one RMF gatherer per z/OS system. With z10 EC provisioning capability (a z-architect function) combined with the Capacity Provisioning Management (CPM), which is a WLM component in z/OS, it is possible in a new, flexible, and automated process to control the activation of On/Off Capacity on Demand (converting PU spares to active PUs and vice versa). Previously, WLM was only able to take CPU capacity from an LP and give it to another LP to decrease CPU delays, which are major cause of key unhappy transactions. Now the CPM (WLM is included in this function) is able to activate spare PUs (CPU, zAAP, and zIIP) through On/Off Capacity on Demand to fix that situation. CPM does not configure reserved or offline logical PUs online. Capacity Provisioning is not performed at the sysplex scope; that is, the z/OS systems where the function will work may belong to a different sysplex.

Background

Unexpected workload spikes may exceed available capacity such that Service Level Agreements cannot be met. Although business need may not justify a permanent upgrade, it might well justify a temporary upgrade.

z10 EC provides an improved and integrated On/Off CoD and CBU concept:

- ▶ Faster activation due to autonomic management and improved robustness

- ▶ Can be partially activated and combined
- ▶ Value Proposition of Capacity Provisioning

Capacity provisioning infrastructure overview

The z/OS provisioning environment, shown in Figure 4-24 on page 169 with all its components, works like this:

- ▶ Each WLM needs to get performance information from other WLMs to make its decisions. However, the WLM conversation cannot be theft XCF because the peers may be out of the same sysplex. Then, the distributed data server (DDS) RMF function is used. WLM metrics (as resource delays and performance index) are available through existing interfaces and are reported through RMFGAT, with one per z/OS. RMF, using its DDS facility through the network, sends all data to a RMF focal point (RMF DDS, in the figure).
- ▶ RMF Common Interface Model (CIM) providers and associated CIM models publish the RMF Monitor III data. CPM observes z/OS systems by connecting to the CIM servers on these systems, and uses these connections to retrieve the required capacity and performance metrics.

CIM is a standard data model developed by a consortium of major hardware and software vendors (including IBM) known as the Distributed Management Task Force (DMTF), which is a part of the Web Based Enterprise Management (WBEM) initiative. It includes a set of standards and technologies that provide management solutions for distributed network environment. CIM creates an interface (API) where applications can, for example, ask system management questions such as: how many jobs are running in the system? Which is the average CPU utilization? These queries must be converted to an API known by the running operating system. In z/OS, CIM is implemented through the Common Event Adapter (CEA) address space.

Capacity Provisioning Manager

Then, the Capacity Provisioning Manager (CPM), a function inside z/OS, retrieves critical metrics from one or more z/OS systems through the CIM structures and protocol. Depending on such metrics, CPM communicates to (local or remote) support elements and HMCs, respectively, via the SNMP network protocol to access On/Off Capacity on Demand. Control over the Provisioning Infrastructure is executed by the CPM through Capacity Provisioning Policy (CPP), which controls the Capacity Provisioning Domain (CPD). A CPM command user interface is through the z/OS system console; however, it is not possible to change the configuration and rules from a console.

The Capacity Provisioning Control Center (CPCC) and the console have no common commands and the CPCC has only a partial use of commands. CPP is created and managed by the software component Capacity Provisioning Control Center, which resides on a Microsoft® Windows® workstation. It provides a system programmer front-end to administer a capacity provisioning policy. CPCC is a GUI component. The policies are not managed by WLM, although they are kept in the WLM couple data set. CPCC is not required for regular CPM operation.

Demand for zAAP processors can only be recognized if at least one zAAP is already online to the system. Demand for zIIP processors can only be recognized if at least one zIIP is already online to the system.

The additional physical capacity provided through CPM is distributed through LPAR and the z/OS systems. In general the additional capacity is available to all LPs, but facilities such as defined capacity (soft capping) or LPAR capping can be used to control the use of capacity.

4.24 Capacity provisioning domain

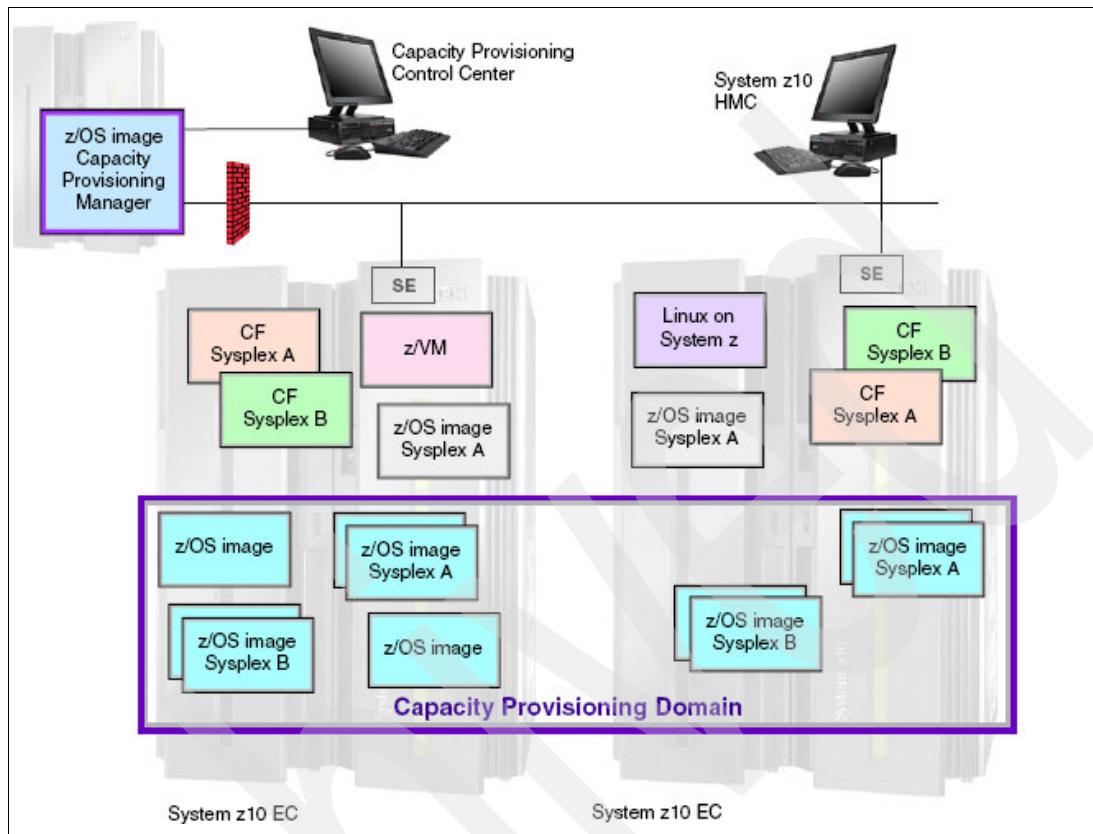


Figure 4-24 Capacity Provisioning domain

Capacity Provisioning domain

Capacity Provisioning Domain (CPD) represents the set of logical partitions and their CECS that are controlled by CPM the Capacity Provisioning Manager. An example is shown in Figure 4-24. The HMCs (driver level D73G or later) of the CECs within a CPD must be connected to the same processor LAN. Parallel Sysplex members can be part of a CPD. There is no requirement that all z/OS members of a Parallel Sysplex must be part of the CPD, but participating z/OS members must all be part of same CPD.

Administrators work through a CPCC interface to define domain configurations and provisioning policies, but use of this interface is not needed during production. Network (SNMP) connectivity is from your hosting system to the HMC or SE.

CPM operates in four different modes, allowing for different levels of automation:

- ▶ Manual mode

This is a command-driven mode; there is no CPM policy active.

- ▶ Analysis mode

In this mode, CPM processes Capacity Provisioning policies and informs the operator when a provisioning or deprovisioning action would be required according to policy criteria.

The operator decides whether to ignore the information or to manually upgrade or downgrade the system using the HMC, the SE, or available CPM commands.

- ▶ Confirmation mode

In this mode, CPM processes Capacity Provisioning policies and interrogates the installed temporary offering records. Every action proposed by the CPM needs to be confirmed by the operator.

- ▶ Autonomic mode

This mode is similar to confirmation mode, but no operator confirmation is needed.

Archived

4.25 Capacity provisioning policy

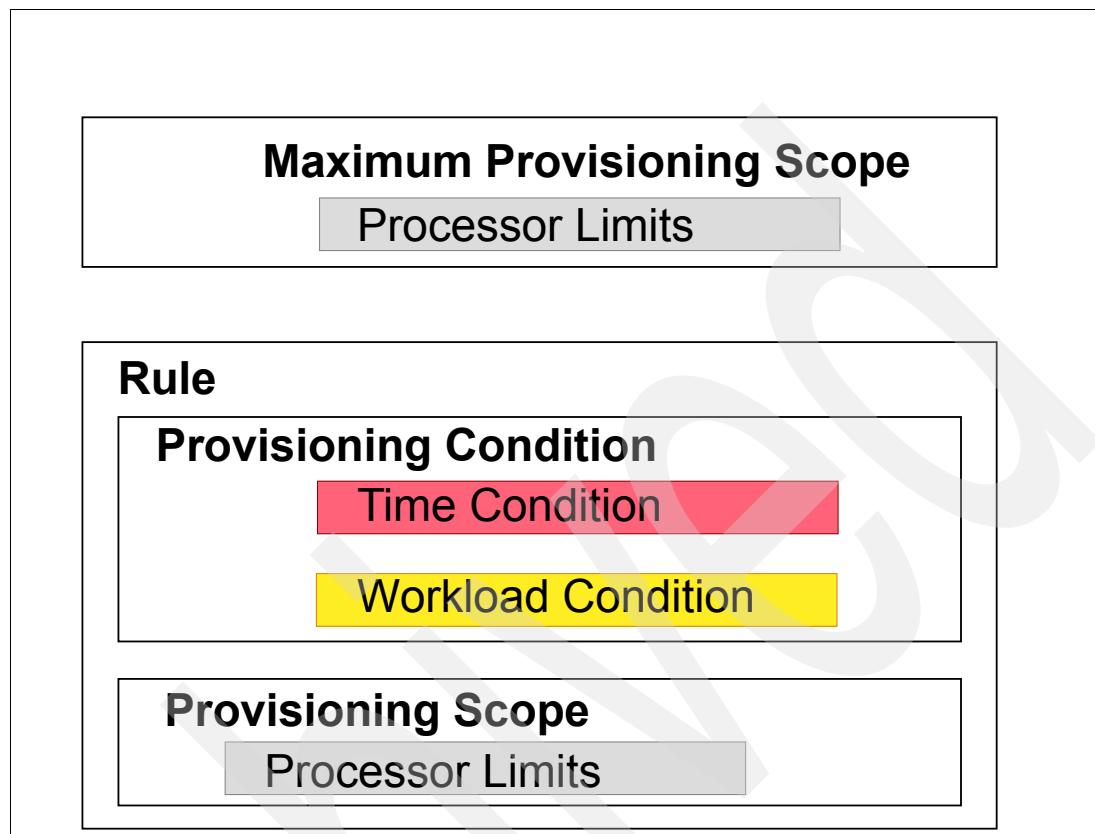


Figure 4-25 Capacity Provisioning policy

Capacity Provisioning policy

The provisioning policy, illustrated in Figure 4-25, defines the circumstances under which additional capacity may be provisioned.

There are three elements in the criteria, as explained here:

- ▶ When provisioning is allowed - time condition:
 - Start time - indicates when provisioning can begin
 - Deadline - provisioning of additional capacity no longer allowed
 - End time - deactivation of additional capacity should end
- ▶ Which work qualifies for provisioning, parameters include - workload condition:
 - The z/OS systems that may execute eligible work
 - Importance filter - eligible service class periods, identified by WLM importance
 - Performance Indicator (PI) criteria:
 - Activation threshold - PI of service class periods must exceed the activation threshold for a specified duration before the work is considered to be suffering
 - Deactivation threshold - PI of service class periods must fall below the deactivation threshold for a specified duration before the work is considered to no longer be suffering
 - Included Service Classes - eligible service class periods

- Excluded Service Classes - service class periods that should not be considered
- ▶ How much additional capacity may be activated expressed in MSU - provisioning scope:
 - Specified in MSUs, number of zaps, and number of zips - one specification per CPC that is part of the Capacity Provisioning Domain

Archived

4.26 WLM buffer pool management

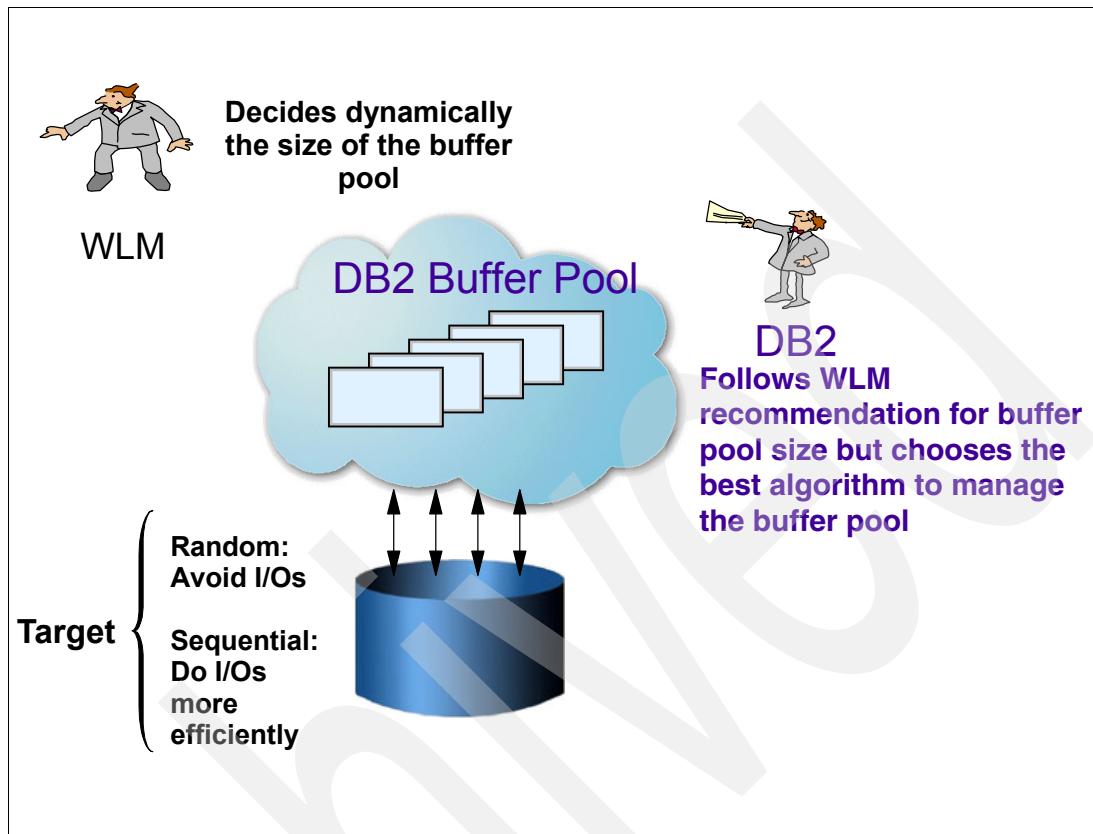


Figure 4-26 Buffer pool management

Buffer pool management

A buffer pool is a set of same-size I/O buffers in virtual storage. Each buffer may contain an I/O physical block (or a control interval in VSAM and DB2). In ECKD™ DASD architecture, the physical block is comprised of a count and a data portion. Usually access methods only keep the data portion in the I/O buffers. Figure 4-26 illustrates buffer pool management.

An I/O buffer is used during an I/O operation as a target for reads or as a source for writes.

A buffer pool has two main design performance objectives:

- ▶ For random reads, if the same data is revisited, you may avoid (save) an I/O operation by having a hit in the buffer pool. A “hit” means that the desired data at the most current level of update is already in one buffer.
- ▶ For random writes, if the logic of the access method (or database) is Store-in (delaying the DASD I/O to a later moment), you may avoid I/O operations because the same data may be updated several times with just one late real I/O operation. Postponing the write I/O operation has another performance advantage in that the requesting task is not placed in the wait state. DB2 has a Store-in implementation for its buffer pools. VSAM has a Store-through, which is the opposite of Store-in—that is, the I/O is done immediately (or synchronously).
- ▶ For sequential reads or writes, by buffering you may have more efficient I/Os by packaging much more data (physical blocks) in a same I/O operation. In this case, “more efficient” means less time per KB of transferred data.

Buffer pool performance

The performance of a buffer pool depends strongly on the following factors:

- ▶ The number of buffers in the buffer pool

For example, a buffer pool for real data and buffer pools for index or meta data, which will help in locating the requested real data

- ▶ The algorithm used to keep data in the buffers of a buffer pool, as explained here:

- Pure random access in a commercial environment - the best of this kind of algorithm, known as Least Recently Used (LRU,) tries to keep in the buffer the population of data that is the most referenced lately.
- Pure sequential access in any environment - the best of this kind of algorithm, also known as “sequential”, immediately flushes the data that was just processed by the application program.
- Mixing between sequential and random - the best algorithm of this kind is a mix of LRU and sequential.

4.27 WLM buffer pool size management

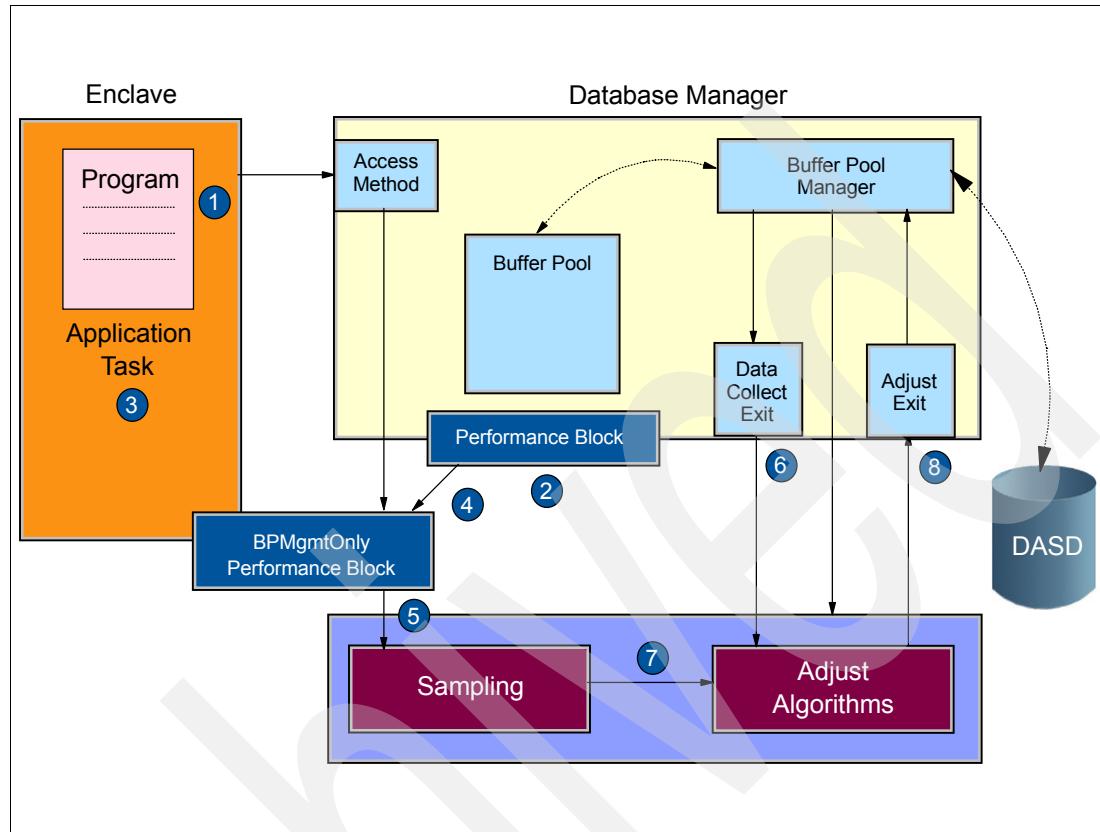


Figure 4-27 WLM buffer pool size management

WLM buffer pool size management

This WLM function is implemented at z/OS V1R8 and exploited by DB2 V9. It allows WLM to dynamically manage DB2 buffer pools. In this case, “managing” refers to deciding at each moment what is the optimum number of buffers in each buffer pool. However, it does not interfere with choosing the right management algorithm; that is still a DB2 decision.

To allow DB2 to exploit such a facility, enter the option YES in the keyword ALTER BUFFERPOOL command: AUTOSIZE (YES/NO). NO is the default.

With YES, DB2 registers the buffer pool (telling the size) with WLM, using a specific API. When the buffer pool starts to become populated by DB2 pages, DB2 informs WLM, in a time basis, about delays caused by the read I/O operation. DB2 also keeps WLM informed about hit ratios for random I/Os. All this information is kept in Performance Blocks.

Figure 4-27 illustrates a chronological sequence of the events associated with WLM buffer pool management.

If WLM determines that a buffer pool is the primary reason for a service class transaction delay causing a PI greater than one, WLM instructs DB2 to increase the size of the buffer pool. It may also compensate for the increase in storage of one pool by reducing the size of another. All changes are made in increments no longer than plus or minus 25%. Only the size of the buffer pool is modified; no other buffer pool characteristics are affected. Any changes made this way to a buffer pool size are maintained across DB2 restarts.

WLM is the right code to implement buffer pool size management because it has a broad view of system performance and the correlated goals. If you are DB2 page fixing any pools, make sure you take that into account before enabling this function. Otherwise, you could end up page fixing as much as 25% more than you originally planned if WLM decides that a buffer pool size should be increased.

Archived

4.28 WLM interfaces

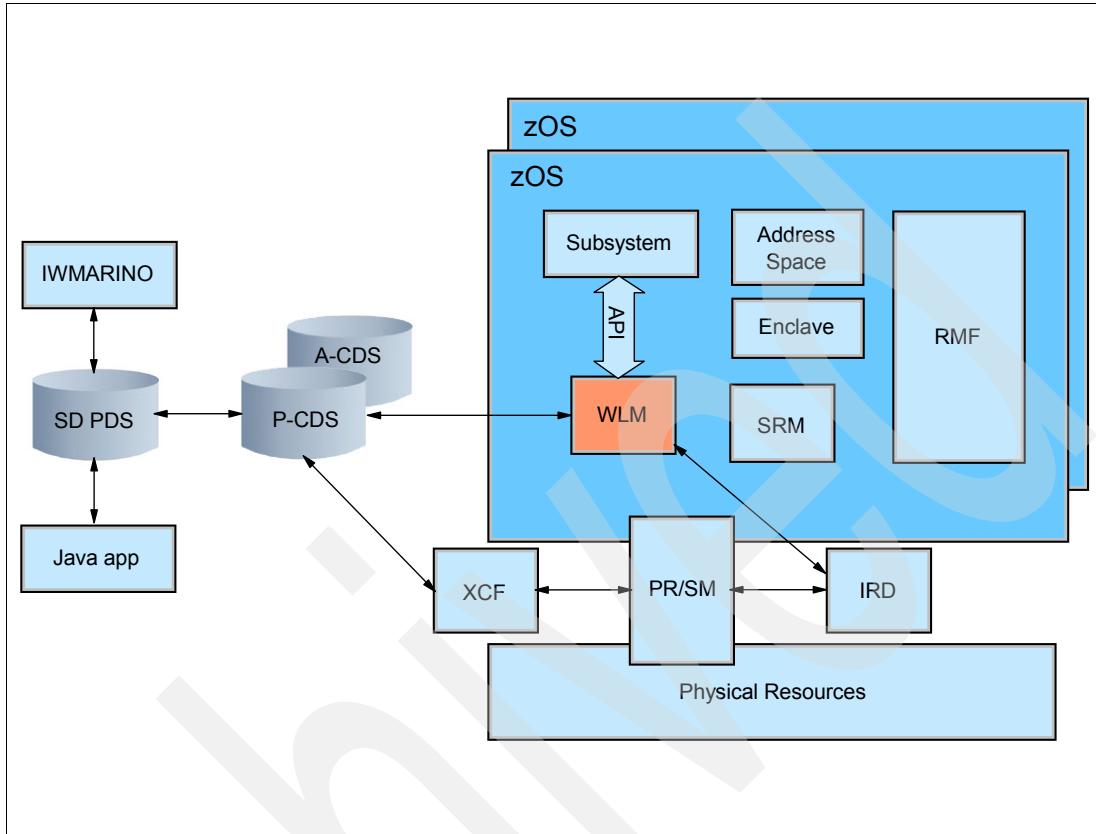


Figure 4-28 WLM most common interfaces

WLM most common interfaces

Figure 4-28 shows the primary objects and interfaces through which WLM acts.

- The administrator's interface is at the left: IWMARINO (the ISPF application) provides access to a PDS member that contains a service definition (SD). Using either IWMARINO or the batch utility, a new service definition can be installed; that is, copied into the Primary Control Data set (P-CDS) for which the Alternate Control Data set (A-CDS) is a backup. There is a new user interface, a Java™ application, that runs on a workstation.
- The SD applies to the entire sysplex. Therefore, at install time, the SD is also copied to each WLM in each z/OS. WLM communicates with other WLMs in a sysplex through the Cross-Coupling Facility (XCF) component.
- WLM, LPAR, SAP, and IOS communicate among themselves to move resources between LPs as necessary to optimize achievement of the performance goals. This happens if IRD is configured and active. In this case there is a need to allocate a CF structure to speed such communication.
- WLM communicates with the work manager subsystems (JES, CICS, DDF, and so on) using over 70 different service calls that together comprise the WLM API set. For example, the IWMCLSFY service is used to classify an incoming unit of work; that is, assign it to a service class and optionally, a report class. There is also the old SRM Sysevent API, which is still valid for certain functions.
- WLM's sampling function periodically (every 0.25 seconds) visits the Task Control Blocks (TCBs) and Service Request Blocks (SRBs) in every Address Space Control Block

(ASCB) and ENCB (Enclave Control Block). For transaction managers that support transaction response time measurement via Performance Block (PB), WLM's sampler also visits every PB.

- ▶ Performance monitors such as RMF, MainView, CMF, Omegamon and others read (through IWMRCOLL API) WLM performance information. This API produces very rich collected performance data. WLM also reads some of RMF's collected data (in the Capacity Provisioning function).

Archived

4.29 WLM console commands (1)

```
D XCF,COUPLE,TYPE=WLM
IXC358I 10.42.54 DISPLAY XCF 501
WLM COUPLE DATA SETS
PRIMARY   DSN: SYS1.XCF.WLM02
            VOLSER: SBOX73      DEVN: 3A3A
            FORMAT TOD          MAXSYSTEM
            05/28/2001 20:26:14      4
            ADDITIONAL INFORMATION:
            NOT PROVIDED
ALTERNATE  DSN: SYS1.XCF.WLM03
            VOLSER: SBOX74      DEVN: 3E3A
            FORMAT TOD          MAXSYSTEM
            05/28/2001 20:26:14      4
            ADDITIONAL INFORMATION:
            NOT PROVIDED
WLM IN USE BY ALL SYSTEMS
```

Figure 4-29 WLM console commands output

WLM console commands

This section presents MVS console display commands that, when entered by the operator or by an automation product, show WLM information data in a sysplex.

- ▶ Enter the following command to display couple data set status. The output from this command is displayed in Figure 4-29.
Display XCF,Couple,Type=WLM
- ▶ Enter the following command to display the active policy, the WLM system state, and the application environment and scheduling environment states. The output from this command is displayed in Figure 4-30 on page 180.
DISPLAY WLM,SYSTEMS

```
D WLM,SYSTEMS
IWM025I 10.46.31  WLM DISPLAY 504
    ACTIVE workload MANAGEMENT SERVICE POLICY NAME: WLMPOL
    ACTIVATED: 2005/02/15 AT: 06:49:24 BY: VAINI      FROM: SC65
    DESCRIPTION: WLM Service Policy
    RELATED SERVICE DEFINITION NAME: Sampdef
    INSTALLED: 2005/02/15 AT: 06:45:45 BY: VAINI      FROM: SC65
    WLM VERSION LEVEL: LEVEL013
    WLM FUNCTIONALITY LEVEL: LEVEL013
    WLM CDS FORMAT LEVEL: FORMAT 3
    STRUCTURE SYSZLWM_transactionUNIT STATUS: CONNECTED
    STRUCTURE SYSZLWM_6A3A2084 STATUS: CONNECTED
    *SYSNAME* *MODE* *POLICY* *workload MANAGEMENT STATUS*
    SC63     GOAL    WLMPOL   ACTIVE
    SC64     GOAL    WLMPOL   ACTIVE
    SC65     GOAL    WLMPOL   ACTIVE
    SC70     GOAL    WLMPOL   ACTIVE
```

Figure 4-30 Display WLM, Systems command output

4.30 WLM console commands (2)

```
D WLM
IWM025I 10.53.35 WLM DISPLAY 516
ACTIVE WORKLOAD MANAGEMENT SERVICE POLICY NAME: WLMPOL
ACTIVATED: 2005/02/15 AT: 06:49:24 BY: VAINI FROM: SC65
DESCRIPTION: WLM Service Policy
RELATED SERVICE DEFINITION NAME: Sampdef
INSTALLED: 2005/02/15 AT: 06:45:45 BY: VAINI FROM: SC65
WLM VERSION LEVEL: LEVEL013
WLM FUNCTIONALITY LEVEL: LEVEL013
WLM CDS FORMAT LEVEL: FORMAT 3
STRUCTURE SYSZWLM_WORKUNIT STATUS: CONNECTED
STRUCTURE SYSZWLM_6A3A2084 STATUS: CONNECTED
```

Figure 4-31 Display WLM command output

Display WLM command

Enter the following command to display the sysplex active policy, the related service definition name, the WLM and CDS levels, and the status of its structures in the Coupling Facility. The output from this command is displayed in Figure 4-31 on page 181.

```
DISPLAY WLM
```

4.31 Other WLM-related console commands

The WLM VARY console command can be used to activate one policy in the sysplex.

VARY WLM,POLICY=OVERRG

IWM001I WORKLOAD MANAGEMENT POLICY OVERRG
NOW IN EFFECT

The WLM Modify console command can be used to activate locally a scheduling environment resource.

MODIFY WLM,RESOURCE=SC63,ON

IWM039I RESOURCE SC63 IS NOW IN THE ON STATE

Figure 4-32 Vary and Modify WLM console commands

Vary and modify WLM commands

The **VARY WLM** command has a sysplex scope that can be used to activate a WLM policy, as shown in Figure 4-32, or an application environment (dynamic or not). See 4.8, “Application environment” on page 142, for more information about this topic.

`VARY WLM,APPLENV=applenvname, or VARY WLM,DYNAPPL=applenvname`

The **MODIFY WLM** command is a local z/OS command that can be used to change the state of a scheduling environment resource.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 183. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *System Programmer’s Guide to: Workload Manager*, SG24-6472
- ▶ *z/OS Intelligent Resource Director*, SG24-5952

Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS MVS Planning: Workload Management*, SA22-7602
- ▶ *z/OS MVS Programming: Workload Management Services*, SA22-7619

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived



Redbooks

ABCs of z/OS System Programming Volume 12

(0.2"spine)
0.17" <-> 0.473"
90<->249 pages



ABCs of z/OS System Programming

Volume 12



Workload Manager, WLM policy

WLM goal management, WLM functions

WLM ISPF application

Installations today process different types of work with different response times. Every installation wants to make the best use of its resources, maintain the highest possible throughput, and achieve the best possible system responsiveness. You can realize such results by using workload management. This IBM Redbooks publication introduces you to the concepts of workload management utilizing Workload Manager (WLM). Workload Manager allows you to define performance goals and assign a business importance to each goal. You define the goals for work in business terms, and the system decides how much resource, such as CPU and storage, should be given to the work. The system matches resources to the work to meet those goals, and constantly monitors and adapts processing to meet the goals. This reporting reflects how well the system is doing compared to its goals, because installations need to know whether performance goals are being achieved as well as what they are accomplishing in the form of performance goals.

The ABCs of z/OS System Programming is a thirteen-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you would like to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks