

Mainframe Internet Integration

**Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth**

SS2012

Work Load Management Teil 1

Übersicht

Literatur

Eine sehr gute Übersicht über den Work Load Manager ist zu finden in Kapitel 6 des Lehrbuches
M. Teuffel, R. Vaupel: „Das Betriebssystem z/OS und die zSeries“. Oldenbourg 2004., ISBN 3-486-27528-3.

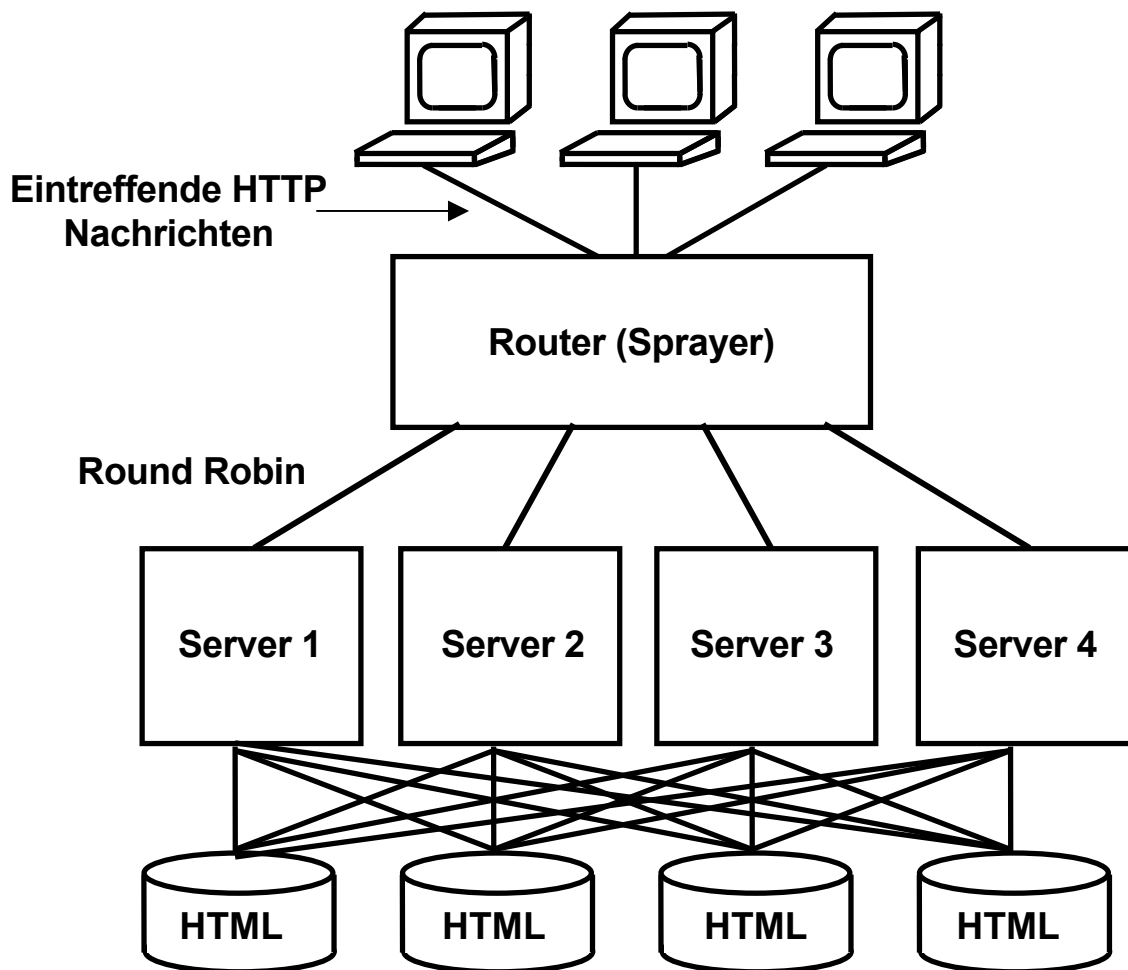
Die beiden wichtigsten Redbooks sind

z/OS Intelligent Resource Director, August 2001, SG24-5952-00

System Programmer's Guide to: Workload Manager, January 2006, SG24-6472-02

Weiterführende Literatur

- z/OS V1.6 Workload Management, Server Limits für DB2 Application Environments:
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IEA2W150/13.3?DT=20040709143734
- DB2 for z/OS Stored Procedures: Through the CALL and Beyond:
<http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sq247083.html>
- USING WORKLOAD (WLM) and DB2 STORED PROCEDURES (SPAS) - Some clarifying Questions and Answers:
<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD103105>
- DB2 stored procedure performance considerations, part two:
<http://www-306.ibm.com/software/tivoli/features/ccr2/ccr2-2004-04/features-db2stored.html>
- Setting up the WLM application environment for interpreted Java routines:
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=/com.ibm.db2.doc.java/bjnkmsr128.htm>
- Workload management (WLM) tuning tips for z/OS Questions and Answers:
http://publib.boulder.ibm.com/infocenter/wasinfo/v5r1//index.jsp?topic=/com.ibm.websphere.zseries.doc/info/zseries/ae/rprf_tunezwlm.html
- z/OS V1.6 Workload Management, Server Limits für DB2 Application Environments:
http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/IEA2W150/13.3?DT=20040709143734
- DB2 for z/OS Stored Procedures: Through the CALL and Beyond:
<http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sq247083.html>
- Workload Manager
http://de.wikipedia.org/wiki/Workload_Manager
- ABCs of z/OS System Programming Volume 12
<http://www.redbooks.ibm.com/abstracts/sq247621.html>
- M. Bensch, D. Brugger, P. Baeuerle, W. Rosenstiel, M. Bogdan, W. Spruth:
"Self-Learning Prediction System for Optimisation of Workload Management in a Mainframe Operating System"
International Conference on Enterprise Information Systems, pp. 212-218, Funchal, Portugal, 12.-26. Juni 2007
<http://www-ti.informatik.uni-tuebingen.de/~spruth/Mirror/iceis.pdf>



Work Load Management für einen WWW Cluster

Angenommen eine Gruppe von Web Servern, die nur Anfragen bearbeiten, z.B. Google. Bei großen Providern kann die Web Site aus hunderten oder tausenden von Servern bestehen

Vorteil: Einheitliche Anwendung, nur Lesezugriffe zu den HTML Daten.

In diesem einfachen Fall verteilt ein vorgeschalteter Router (auch als Sprayer bezeichnet) die eintreffenden Anfragen nach dem Round Robin Verfahren auf die einzelnen Server,

Die Google Workload Aufteilung ist trivial einfach, weil alle Anfragen identisch sind und nur wenige unterschiedliche Server-seitige Verarbeitungsprogramme benötigt werden. Alle Datenbankzugriffe sind Lesezugriffe, und eine **Datenintegrität** der auf mehreren duplizierten Datenbank abgespeicherten Daten **ist nicht erforderlich**.

z/OS Work Load Management Übersicht (1)

Die folgenden drei Seiten enthalten eine sehr schöne Übersicht über das z/OS Work Load Management aus Wikipedia (http://de.wikipedia.org/wiki/Workload_Manager). Wir gehen später auf die hier erwähnten Eigenschaften im Detail ein.

Die in jedem Augenblick von einem z/OS System auszuführende Datenverarbeitung wird als „Arbeit“ (Work) bezeichnet. Der einem bestimmten Prozess zuzuordnende Teil davon wird als Arbeitseinheit (work unit) bezeichnet. Die in einem bestimmten Augenblick anfallende „Work Load“ ist die Summe aller Arbeitseinheiten.

Der Workload Manager (WLM) ist die z/OS Betriebssystemkomponente, die für die Arbeit auf dem Rechner den Zugang zu den Betriebsmitteln steuert. Auf einem Großrechner ist ein prioritisierter Zugang zu Betriebsmitteln notwendig, da viele unterschiedliche Anwendungen den Rechner gleichzeitig nutzen und eine den Benutzerwünschen entsprechende Ressourcenverteilung erfolgen muss.

Betriebsmittel sind z.B. CPU Zeit, Hauptspeicherplatz, I/O Kanalkapazität oder Plattenspeicherzugriffe. Auch Netzwerk Adapter oder Locks können Ressourcen sein. WLM steuert die Betriebsmittelvergabe auf der Basis von Dienstklassen (Service Classes). Arbeitseinheiten werden den Dienstklassen über einen Klassifizierungsmechanismus zugeordnet. Die Klassifizierung wird durch den Systemadministrator des z/OS-Systems vorgenommen und kann anhand von Attributen, die für die Programmprodukte unter z/OS existieren, vorgenommen werden. Beispiele für Attribute sind Benutzernamen, Transaktionsnamen, Transaktionsklassen oder Programmnamen, die in den Anwendungen verwendet werden. Als weiteres definiert der Systemadministrator eine Zielvorgabe für die Dienstklassen. Die Zielvorgabe kann die durchschnittliche Antwortzeit der Arbeitseinheiten, die in der Klasse laufen, umfassen, einen prozentualen Anteil der Arbeitseinheiten, die in einer bestimmten Zeit enden sollen, oder eine durchsatzorientierte Vorgabe darstellen. Welches Ziel für eine Dienstklasse vergeben werden kann, hängt davon ab, wie viele Informationen der Workload-Manager über die Anwendungen erhält. Neben der Zielvorgabe wird jeder Dienstklasse eine Wichtigkeit zugeordnet, die festlegt, welche Klassen bevorzugt bzw. benachteiligt werden sollen, wenn die Betriebsmittel im System nicht mehr ausreichend zur Verfügung stehen.

Übersicht (2)

WLM benutzt einen Regelmechanismus, um zur Laufzeit den Zugang zu den Betriebsmitteln zu steuern. Dazu werden kontinuierlich Daten aus dem z/OS-System gesammelt. Dies sind Informationen über die Wartezustände der Arbeitseinheiten auf die Betriebsmittel, die Anzahl der laufenden Arbeitseinheiten und deren Abarbeitungszeiten. Die Informationen werden in Dienstklassen zusammengefasst entsprechend der Klassifizierung, die durch den Systemadministrator vorgenommen wurde. Dann wird auf der Basis dieser Informationen die Zielerfüllung für jede Klasse berechnet und, falls notwendig, der Zugang zu den Betriebsmitteln angepasst. Die Anpassung erfolgt immer in Abhängigkeit von der Wichtigkeit der Klassen und dem Grad in dem das Ziel verfehlt wird. Das heißt, die wichtigste Klasse, die am weitesten ihr vorgegebenes Ziel verfehlt hat, wird als erste betrachtet und die Klassen mit der geringsten Wichtigkeit sind die potenziellen Kandidaten, um Betriebsmittel abzugeben. Dabei wird allerdings berücksichtigt, ob ein potenzieller Spender (Donor) auch tatsächlich das benötigte Betriebsmittel verwendet. Dieser Regelmechanismus läuft typischerweise alle 10 Sekunden unter z/OS ab; in der Zwischenzeit werden die Daten für das nächste Berechnungsintervall gesammelt. Ein Berechnungsintervall endet, wenn eine Anpassung zugunsten einer Dienstklasse durchgeführt werden kann.

WLM steuert den Zugang zu den Prozessoren und I/O-Einheiten des Systems, den Zugang zum Speicher und die Bereitstellung von Adressräumen, um Programme für bestimmte Anwendungen abarbeiten zu lassen. Der Zugang zu den Prozessoren wird zum Beispiel über Dispatch Priorities geregelt. Dazu wird allen Arbeitseinheiten einer Dienstklasse dieselbe Priorität zugeordnet, wobei jedoch die Vergabe dieser Priorität nicht in jedem Fall mit der Definition der Wichtigkeit der Dienstklasse übereinstimmen muss. Vielmehr orientiert sie sich an der aktuellen Auslastung des Systems, den Anforderungen der Klasse und ihrer Zielerfüllung. Dieses Verhalten des z/OS-WLM nennt man auch zielorientiertes Workload-Management, und es ist ein wichtiges Unterscheidungskriterium zu anteilsorientiertem Workload-Management, bei dem feste Zugänge zu den Betriebsmitteln vergeben werden. Letzteres findet sich häufig in Workload-Management-Komponenten von Unix-Systemen.

Übersicht (3)

Der WLM setzt umfangreiche adaptive Algorithmen ein um Rechner zu steuern, so dass Zielvorgaben (Business Goals) des Anwenders erfüllt werden. Im Gegensatz zu den Implementierungen auf anderen Plattformen (z.B. HP Superdome, Sun M9000) erfolgt die hier beschriebene Steuerung voll automatisch, ohne Eingriffe durch den Systemadministrator. Es wird ganz auf den Einsatz von vorgefertigten Regeln und von fest einzustellenden Parametern verzichtet. Die Steuerungsalgorithmen passen sich ebenfalls automatisch an das Umfeld an. Die hierfür erforderlichen Anpassungen an eine sich ständig ändernde Workload erfolgen automatisch mit einer Taktrate von z.B. 10 Sekunden.

Der zweite essentielle Unterschied des z/OS-WLM zu den Workload-Management Implementierungen auf anderen Plattformen ist die starke Verflechtung mit den Anwendungen und Programmprodukten, die unter einem z/OS-Betriebssystem ablaufen. So ist es durch die ständige Kommunikation zwischen dem WLM und diesen Anwendungen möglich, die Eigenschaften der Anwendungen zu erkennen und im System durch den WLM zu steuern. Dies ist bis dato auf keinem anderen System möglich, in denen jedwede Steuerung auf Prozesse begrenzt ist.

Neben der Steuerung eines Systems bietet der z/OS-WLM eine Reihe von Schnittstellen, die es Lastverteilungskomponenten erlauben, Informationen aus dem System zu erhalten, um eine intelligente Verteilung von Arbeit auf eines oder mehrere z/OS-Systeme vorzunehmen. Mehrere z/OS-Systeme können in einem Parallel Sysplex zusammengeschaltet werden, und diese Kombination wird ebenfalls unterstützt, um nach außen ein einheitliches Bild abzugeben. z/OS WLM verfügt außerdem über eine Reihe von weiteren Funktionen, die die Lastverteilung auf einem physischen System zwischen mehreren logischen Systeme unterstützen und den Zugang zu großen Plattenfarmen in Abhängigkeit von der daraufzugreifenden Arbeit steuern.

z/OS Work Load Manager

Der Work Load Manager (WLM) ist ein z/OS Alleinstellungsmerkmal. Andere Plattformen, z.B. HP Superdome, verfügen ebenfalls eine als Work Load Manager bezeichnete Komponente, die vor allem benutzt wird, um virtuelle Maschinen realen CPUs zuzordnen. Der HP WLM hat jedoch nicht den Funktionsumfang des z/OS WLM.

<http://www.redbooks.ibm.com/redbooks/pdfs/sq246472.pdf>

http://h18004.www1.hp.com/products/quickspecs/11726_div/11726_div.HTML

Begriffe

- Work** andere Bezeichnungen Datenverarbeitung. Data Processing Is das, was ein Rechner durchführt
- Work Unit** ein Stück Datenverarbeitung, z.B. Durchführung einer Transaktion oder eines Stapelbearbeitungsjobs
- Work Load** = \sum Work Units , die Summe aller Work Units, die ein rechner in einem gegebenen Zeitpunkt bewältigt
- Resourcen** die Betriebsmittel, über die ein rechner verfügt. Beispiele sind CPU Zyklen, Hauptspeicherplatz, I/O Kanal Kapazität, Plattenspeicherzugriffe
- Service Unit** Ressourcenverbrauch einer Work Unit
- Service Class** (Dienst Klasse), Gruppe von Work Units mit ähnlichen Vorgaben wie Antwortzeit (Response Time), Turn-around-Time, Priorität, usw.
- Goal** andere Bezeichnung Business Goal, Zielvorgabe für für eine Service Class, z. B. 90 % aller Transaktionen einer bestimmten Service Class sollen eine Response Time < 0,3 Sekunden haben.

Traditionelle Work Load Management Verfahren

Es existieren viele Möglichkeiten und Einstellungsalternativen, die Ausführung der Work Load zu steuern. Beispiele sind:

- Programme laufen Run-to-Completion, oder unterliegen einer Zeitscheibensteuerung
- Die Anzahl der gleichzeitig aktiver Prozesse kann vergrößert oder verkleinert werden, um z.B. jedem Prozess eine optimale Größe an realen Speicher zuzuordnen. Prozesse können zeitweise ausgelagert werden, wenn Ressourcen knapp werden.
- Multithreading, Anzahl von Subsystem Instanzen
- In einem Sysplex können Prozesse physischen Servern optimal zugeordnet werden
- I/O Kanäle können den LPARs optimal zugeordnet werden
- Prioritäten können angepasst werden
- usw.

Die Komplexität wächst mit der Systemgröße

Problembereiche sind

- Stapelverarbeitung und interaktive Verarbeitung müssen gleichzeitig laufen
- Belastungsschwankungen treten auf (während des Tages, Woche, Jahr)
- Die Affinität der Prozesse zu ihren Daten soll optimiert werden
- Die Zuordnung der Prozesse zu realen CPUs muss erfolgen
- Es existieren unterschiedliche I/O Anforderungen und I/O Belastungen

In einer IT-Umgebung ohne Mainframe ist es üblich, unterschiedlichen Anwendungen unabhängigen physischen Servern zuzuordnen. Die Folge ist eine schlechte Auslastung der Server (typisch 20 % CPU-Auslastung). Dies erscheint akzeptabel, denn die Hardware ist billiger als bei einer Mainframe Lösung. Jedoch die Folgen sind:

- Viele Server (tausende in großen Unternehmen)
- Heterogene IT-Landschaft
- Komplexe LAN Strukturen
- Hoher Administrationsaufwand
- Hoher Energieverbrauch und Kühlungs-/Klimatisierungsaufwand

Vor allem die hohen Administrationskosten machen diese Lösung zunehmend unattraktiv.

In einer Mainframe Installation sind Energieverbrauch und Kühlungs-/Klimatisierungsaufwand deutlich geringer, da eine CPU Auslastung im 90 – 100% Bereich möglich ist. Dies ermöglicht der z/OS Workload Manager. Weiterhin ist der Administrationsaufwand deutlich geringer, wozu die **Tivoli** und **Unified Resource Manager** Komponenten beitragen.

System Ressourcen

Unter System Ressourcen versteht man Elemente wie

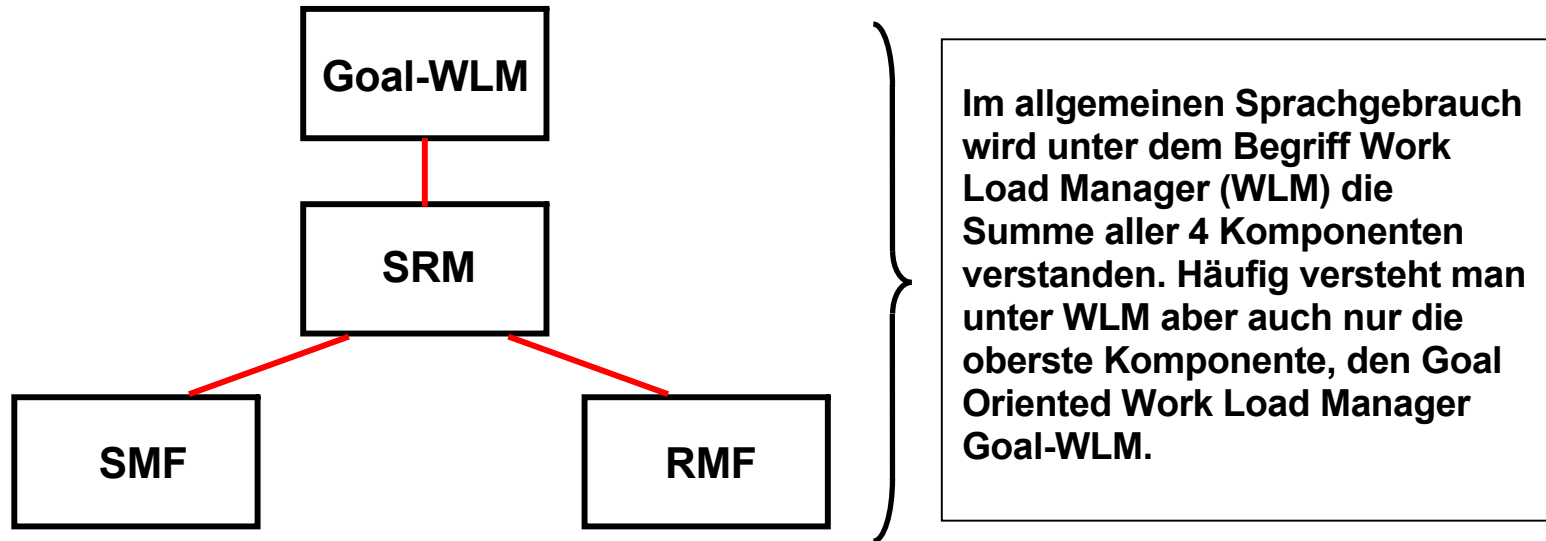
- Verarbeitungskapazität der individuellen CPUs,
- Platz im Hauptspeicher,
- Channel Subsystem Verarbeitungskapazität,
- I/O Kanal Übertragungskapazität,
- Plattenspeicherzugriffe,
- ...

Der Workload Manager hilft, alle System Ressourcen optimal auszunutzen.

Anmerkung: Die deutsche Sprache verwendet die Schreibweise Ressource. Die englische Schreibweise ist Resource. Wir benutzen den Begriff Ressource/Resource teilweise alleinstehend und teilweise als Teil eines Begriffes, z.B. „System Resource Manager“.

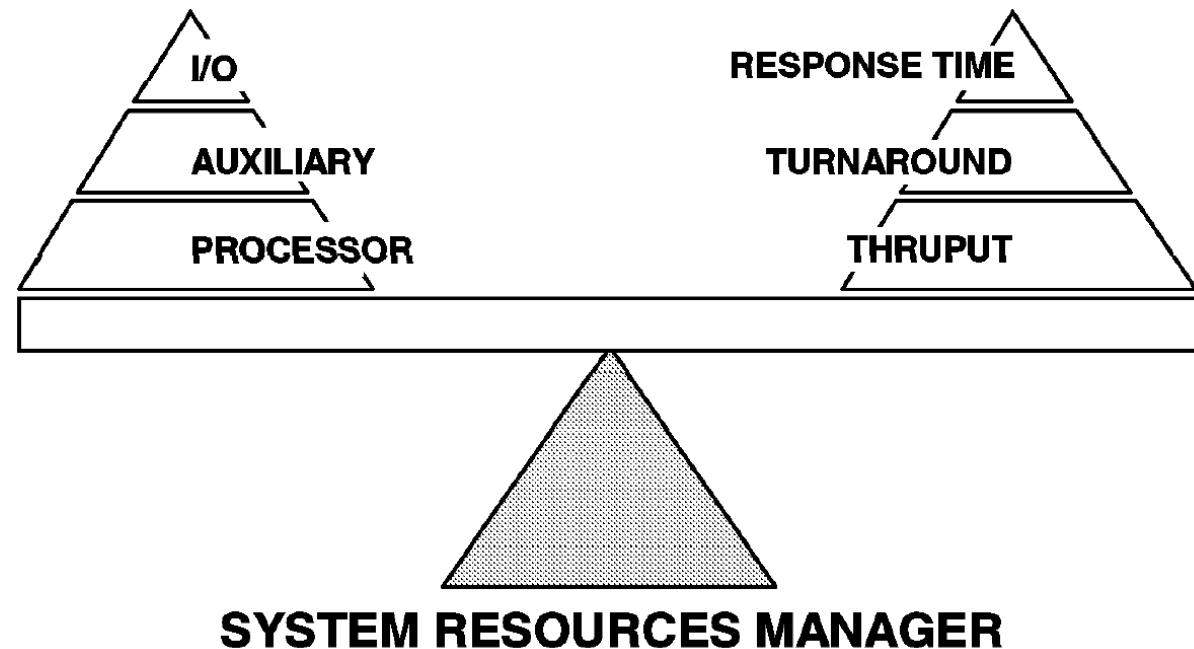
Die unterschiedliche Schreibweise ist verwirrend. Wir verwenden deshalb ausschließlich die englische Schreibweise.

Die Verwendung der deutschen Übersetzung „Betriebsmittel“ ist im Zusammenhang mit WLM eher ungebräuchlich.



Es existieren vier z/OS Work Load Manager Komponenten

- Der **Goal oriented Work Load Manager** Goal-WLM stellt die Schnittstelle zum Systemadministrator dar und erlaubt eine voll-automatische Ablaufsteuerung-
- Der **System Resource Manager** (SRM) ist eine Komponente des z/OS Kernels.
- Die **System Management Facility (SMF)** sammelt System-relevante Information über die Konfiguration, die Auslastung der einzelnen Systemkomponenten (z.B. Hauptspeicher, CPUs, paging/swapping Aktivitäten, I/O Aktivitäten usw. Die Ergebnisse werden in SYS1.MANn Data Sets festgehalten.
- Die **Resource Measurement Facility** (RMF) erstellt Durchschnittswerte über festgelegte Zeitintervalle und macht sie als Input für die erwähnte Tivoli Komponente verfügbar.



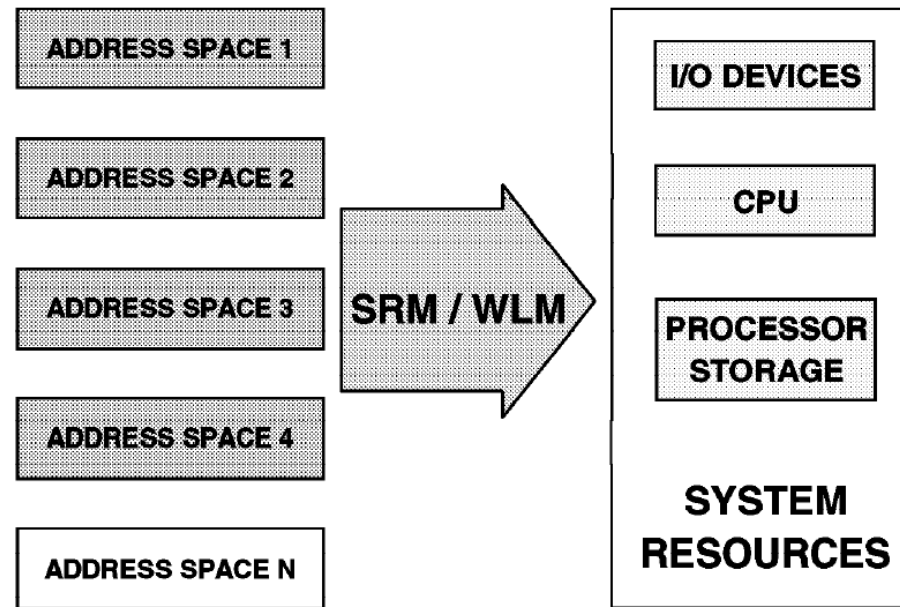
Die System Resource Manager Komponente des Work Load Managers beobachtet für alle angeschlossenen Systeme:

- CPU Auslastung
- Hauptspeicher Nutzung
- I/O Belastung

Der System Resources Manager stellt sicher dass:

- die System Ressourcen möglichst voll genutzt, aber nicht übercommitted werden, und
- alle Benutzer einen fairen Anteil der System Resource erhalten.

SRM überwacht die Nutzung der Ressourcen und ordnet sie periodisch neu zu. Wenn eine Resource schlecht ausgelastet ist, versucht SRM die Systembelastung zu erhöhen. Wenn eine Resource überlastet ist, versucht SRM die Systembelastung zu verringern.



System Resource Manager

Prozesse sind entweder aktiv (ihnen ist ein Adressenraum mit realem Hauptspeicherplatz zugeordnet), oder sie sind inaktiv (auf einen Plattenspeicher ausgelagert).

SRM bestimmt, welchen aktiven Prozessen (bzw. ihren Adressenräumen) welche System Ressourcen (und wieviel davon) zugeordnet werden. SRM entscheidet dies um zwei grundsätzlich miteinander konkurrierende Ziele zu erreichen:

1. System Resources werden einzelnen Prozessen zugeteilt um die für die Installation festgelegten Ziele in Bezug auf Antwort-Zeit, turnaround-Zeit und Prioritätsanforderungen optimal zu erfüllen
2. System Ressourcen werden optimal genutzt um einen möglichst hohen Systemdurchsatz zu erreichen.

System Management Facility SMF

Die System Management Facility (SMF) überwacht kontinuierlich, welche Betriebsmittel eine Work Unit (eine Gruppierung von Anwendungen mit ähnlichen Anforderungen) 1. benutzt und 2. auf welche sie wartet.

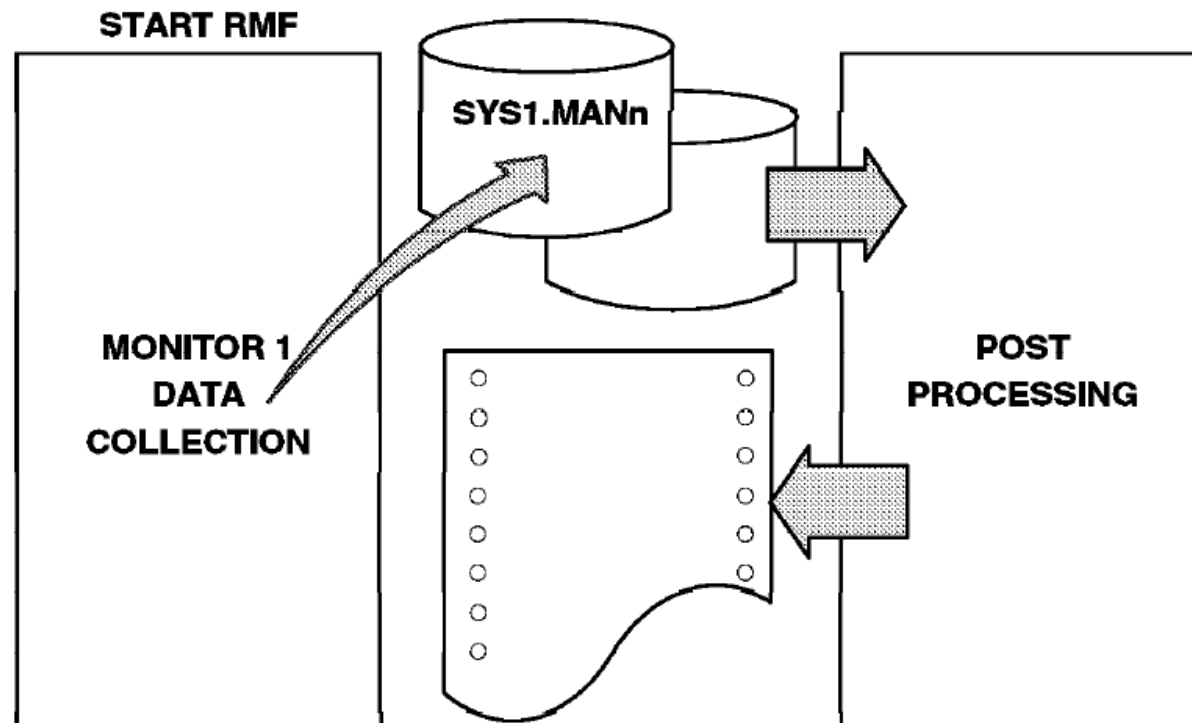
SMF sammelt Daten über den aktuellen Zustand aller verwalteten Betriebsmittel. Dies sind Information über

- die Prozessoren,**
- den Speicher und**
- die Nutzung der Platteneinheiten und Kanäle.**

Aus den gesammelten Daten wird für jede Work Unit festgestellt, ob sie Betriebsmittel benutzt oder darauf wartet. Der Workload-Manager verwendet diese Daten, um den Zugang der Service-Klassen zu den Betriebsmitteln zu regeln.

Die System Management Facility (SMF) bewirkt, dass System-bezogene Information über die Konfiguration, die Workload und paging/swapping Aktivitäten in SYS1.MANn Data Sets gesammelt werden.

SMF Records des Job Entry Subsystems werden ebenfalls in den SYS1.MANn Data Sets gespeichert. Diese Records enthalten Information über die Start- und Stopzeiten, Processor Aktivitäten und Hauptspeichernutzung für jeden Job Step und jede TSO Session. Hilfsprogramme analysieren die Daten und erstellen Berichte.



RMF Monitor

Die Resource Measurement Facility (RMF) ist ein eigener Prozess in einem eigenen Adressenraum.

Ein RMF Monitor sammelt System Daten, z.B. CPU Auslastung, DASD Aktivitäten, Belastung der Kanäle, Hauptspeicher Auslastung und Nutzung sowie andere Aktivitäten. RMF erstellt Zusammenfassungen und Berichte hierüber. Der System Administrator bestimmt die Größe der Messintervalle.

Goal oriented Work Load Manager

Der System Resource Manager ist auf statische Parameter des System Administrators angewiesen und kann sich nur in geringem Umfang an dynamische Änderungen anpassen.

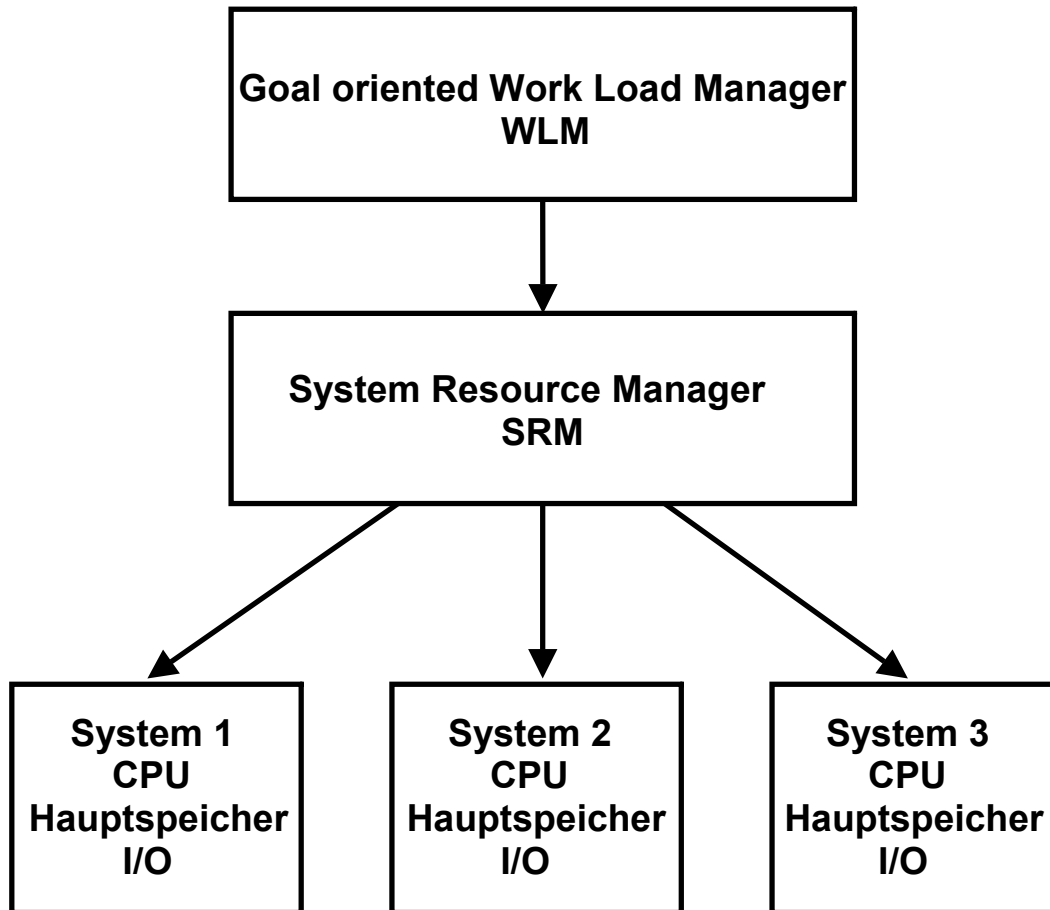
Der Goal Oriented Work Load Manager verfügt über umfangreiche adaptive Algorithmen um die Betriebsmittel abhängig von der sich dynamisch ändernden Systemlast zu steuern, ohne erforderliche Eingriffe durch den Systemadministrator..

Der z/OS Goal oriented Work Load Manager (Goal-WLM) erweitert den Funktionsumfang des SRM, und dehnt ihn auf den ganzen Sysplex aus. Er stellt eine Shell für den System Resource Manager (SRM) dar. Goal-WLM ist eine SRM Erweiterung, welche die existierenden SRM Mechanismen für die Zuordnung von Ressourcen zu Work Units um Performance Management Funktionen erweitert.

Goal oriented Workload Management stellt eine Veränderung der Fokussierung dar, von

- **tuning auf der System Resource Ebene, nach**
- **Definitionen des erwarteten Leistungsverhaltens.**

Diese Eigenschaften sind ein z/OS Alleinstellungsmerkmal und in keinem anderen Betriebssystem vorhanden.



Angenommen ein Sysplex mit drei Systemen.

Die System Resource Manager Komponente des Work Load Managers beobachtet für alle angeschlossenen Systeme:

- CPU Auslastung
- Hauptspeicher Nutzung
- I/O Belastung

Der Goal oriented Work Load Manager steuert die optimale Auslastung der Systeme des Sysplex.

Sysplex Betrieb

Resource Management

Prozesse, die in einem z/OS System laufen, benötigen Betriebsmittel (Ressourcen)

- CPU Zeit
- Hauptspeicherbedarf (Rahmen im Hauptspeicher)
- I/O Operationen

Es existieren große Unterschiede, in welchem Umfang die einzelnen Prozesse Ressourcen benötigen. Manche Prozesse benötigen viel CPU Zeit, aber wenig Hauptspeicherplatz. Bei anderen Prozessen ist es umgekehrt.

Der Ressourcenverbrauch eines Prozesses wird in Service Units gemessen. Es existieren unterschiedliche Service Unit Definitionen für die CPU Zeit, den Hauptspeicherbedarf und die I/O Operationen.

Wir schauen uns das Ressourcenmanagement an Hand der Hauptspeicherverwaltung näher an.

Flattern (Thrashing)

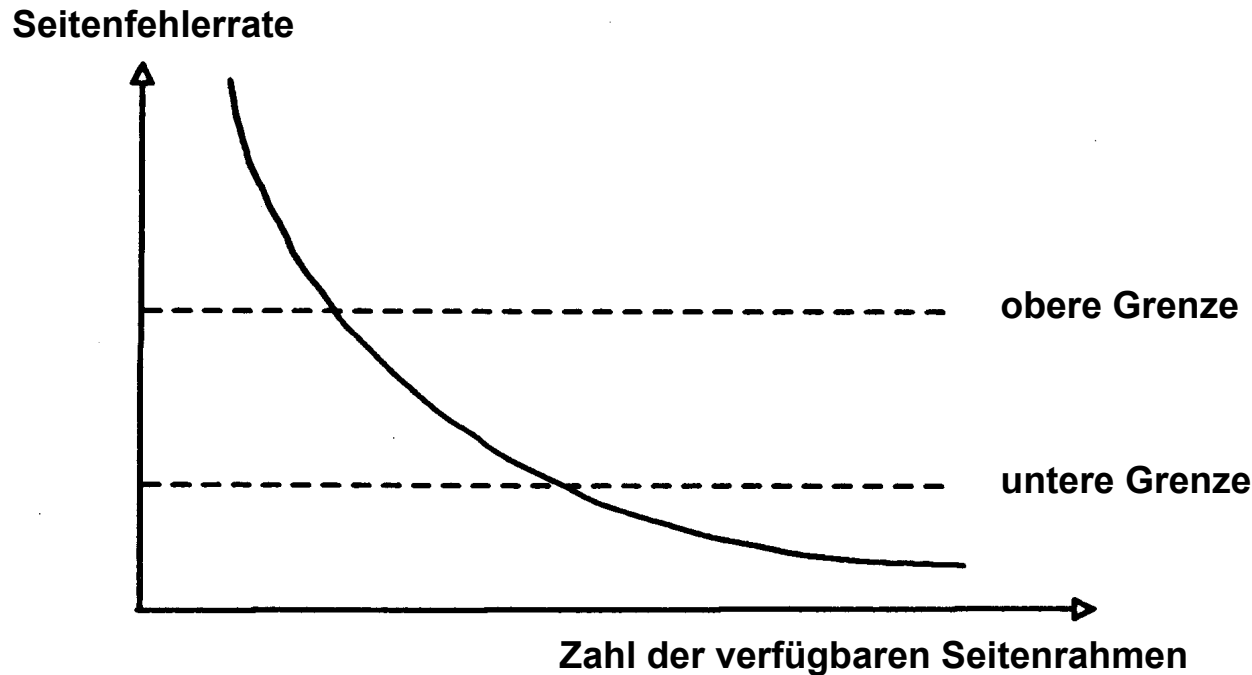
Auf einem Rechner laufen gleichzeitig zahlreiche Prozesse. Sie wechseln zwischen den Zuständen laufend, wartend und ausführbar. Wir bezeichnen diese Prozesse als **aktive** Prozesse (siehe Vorlesung Wintersemester, Verarbeitungsgrundlagen, Teil 1). Normalerweise steigt die Auslastung der CPUs, je größer die Anzahl der aktiven Prozesse ist.

Jeder aktive Prozess beansprucht realen Hauptspeicher für seinen Working Set, mit einer gewissen minimalen Größe (siehe Vorlesung Wintersemester, Verarbeitungsgrundlagen, Teil 2) . Nur ein Teil der Seiten des virtuellen Speichers ist in jedem Augenblick in Rahmen des realen Speichers abgebildet .Der größere Teil der Seiten eines virtuellen Speichers ist in jedem Augenblick auf einem externen Seitenspeicher ausgelagert. Der reale Speicher besteht somit aus 2 Teilen: dem realen Hauptspeicher und dem externen Seitenspeicher (pagefile.sys unter Windows). Beim Zugriff zu einer ausgelagerten Seite (nicht in einem Rahmen des Hauptspeichers abgebildet) erfolgt eine Fehlseitenunterbrechung (Page Fault). Diese bewirkt den Aufruf einer Komponente des Überwachers (Seitenüberwacher, Paging Supervisor), der die benötigte Seite aus dem externen Seitenspeicher holt und in den Hauptspeicher einliest. In den meisten Fällen muss dafür Platz geschaffen werden, indem eine andere Seite dafür auf den externen Seitenspeicher ausgelagert wird. Dieser Vorgang wird als Demand Paging bezeichnet.

Wenn Ihr Windows- oder Apple Rechner genügend viel Hauptspeicher hat, können evtl. 100 % der Seiten eines virtuellen Speichers in Rahmen des realen Hauptspeichers abgebildet werden. Bei einem Mainframe mit bis zu mehreren TByte Hauptspeicher ist dies fast nie der Fall.

Steigt die Anzahl der aktiven Prozesse, so wird der Prozentsatz der Seiten immer kleiner, die für jeden Prozess in Rahmen des realen Hauptspeichers (an Stelle des externen Seitenspeichers) abgebildet werden. Je kleiner der Prozentsatz, um so größer ist die Wahrscheinlichkeit einer Fehlseitenunterbrechung. Die Seitenfehlerrate steigt umso mehr an, je geringer die Anzahl der im Hauptspeicher verfügbaren Rahmen ist.

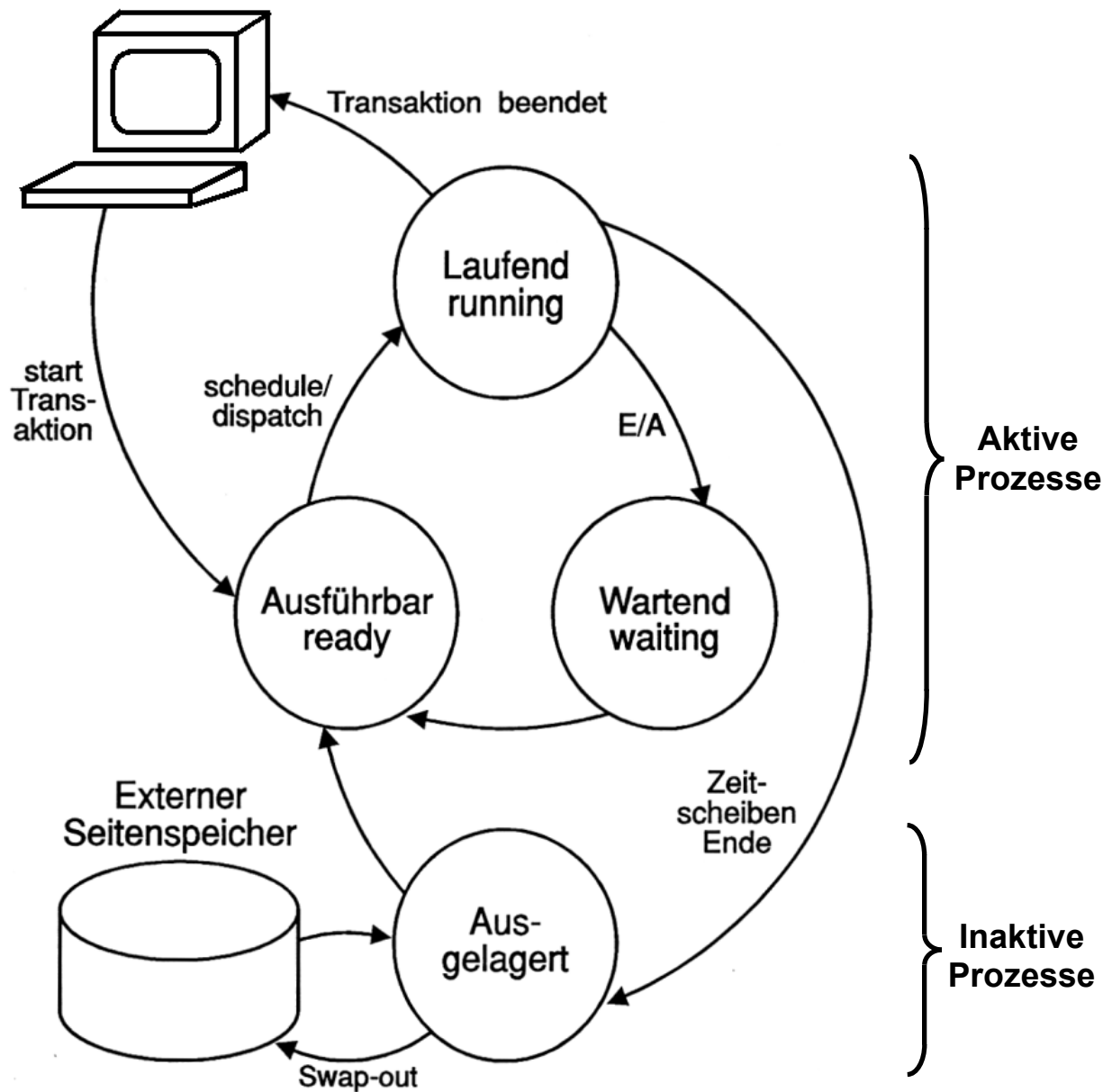
Wird die Anzahl zu gering, tritt ein als „Flattern“ (Thrashing) bezeichnetes Problem auf. Die Prozesse stehen sich gegenseitig benötigte Rahmen und können auf Grund zahlreicher Fehlseitenunterbrechungen kaum noch produktive Arbeit leisten.



Begrenzung der Seitenfehlerrate

Um Flattern zu verhindern, beobachtet WLM ständig die Anzahl der Fehlseitenunterbrechungen pro Zeitintervall (z.B. alle 10 Sekunden). Übersteigt die Anzahl der Fehlseitenunterbrechungen eine festgelegte obere Grenze, reduziert WLM die als „**Multiprogramming Level**“ bezeichnete Anzahl der aktiven Prozesse. Einige Prozesse werden inaktiviert, indem sie mit einem als Swap-Out bezeichneten Verfahren ganz auf den externen Seitenspeicher ausgelagert werden; sie verlieren alle Rahmen im realen Hauptspeicher. Den übrigen aktiven Prozessen stehen damit mehr Hauptspeicherrahmen zur Verfügung, und die Seitenfehlerrate sinkt. Es werden solange Prozesse mit niedriger Priorität inaktiviert, bis die Seitenfehlerrate auf einen akzeptablen Wert sinkt.

Unterschreitet die Seitenfehlerrate eine festgelegte untere Grenze, mag es sinnvoll sein, die Anzahl der aktiven Prozesse wieder zu erhöhen (Swap-in) um die Auslastung der CPUs und/oder anderer Ressourcen zu verbessern.



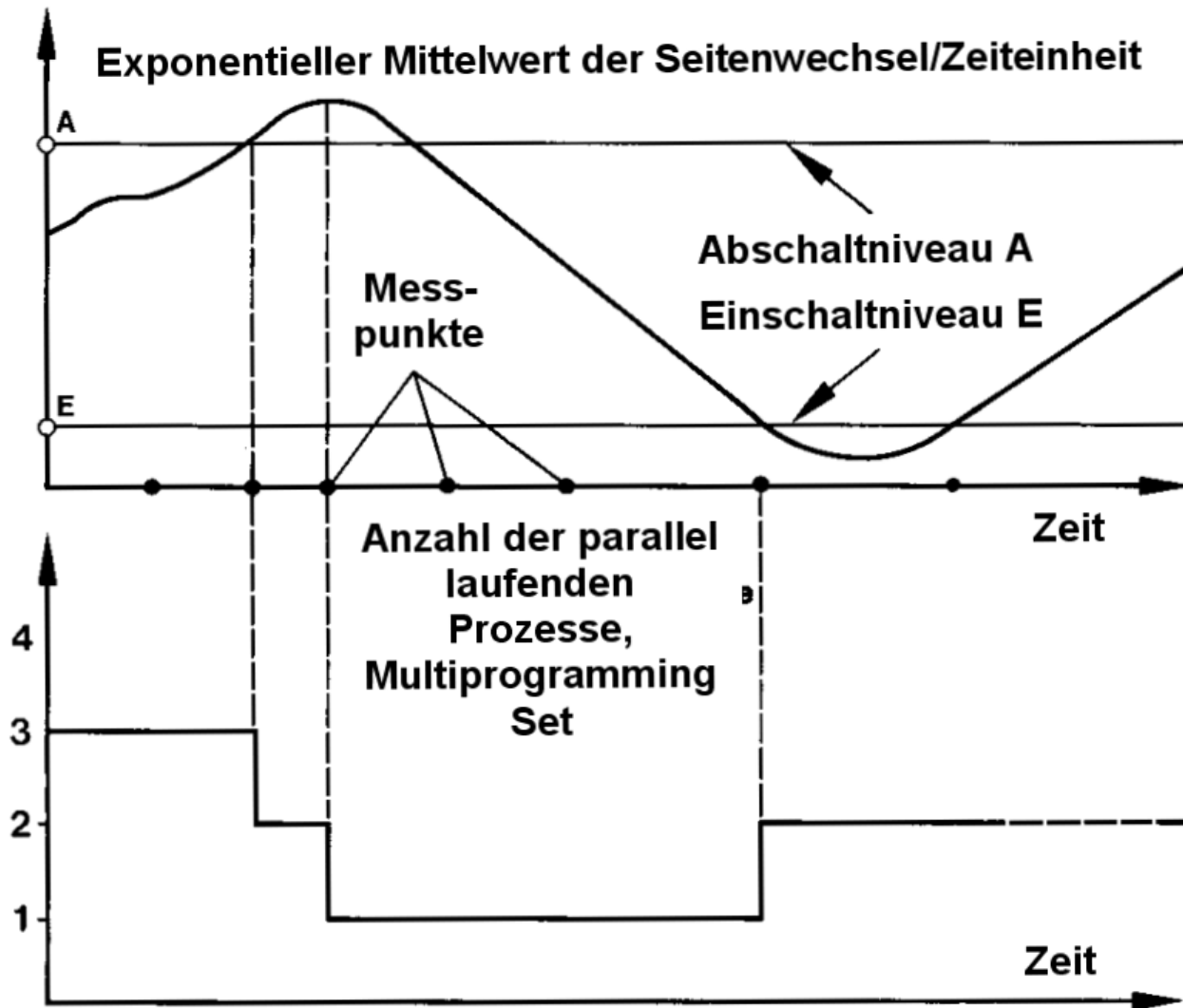
Aktive Prozesse beanspruchen Rahmen im realen Hauptspeicher.

Inaktive Prozesse sind auf den externen Seitenspeicher (auxiliary Storage) ausgelagert und verfügen über keine Rahmen im realen Hauptspeicher.

Verringert man die Anzahl der aktiven Prozesse indem man die Anzahl der (ausgelagerten) inaktiven Prozesse erhöht, stehen den restlichen aktiven Prozessen mehr Rahmen im realen Hauptspeicher zur Verfügung. Die Seitenfehlerrate sinkt.

Wenn die Anzahl der aktiven Prozesse zu klein ist, werden möglicherweise andere Ressourcen, z.B. CPU Zeit schlecht ausgenutzt.

Auslagern von Prozessen



Als Multiprogramming Set wird die Menge der derzeit aktiv Prozesse bezeichnet.

Dargestellt ist, wie über festgelegte Zeitintervalle die durchschnittlich Seitenfehlerrate ermittelt wird.

Übersteigt die durchschnittliche Seitenfehlerrate den Grenzwert A, so wird die Multiprogramming Level (Anzahl der Prozesse im Multiprogramming Set) reduziert.

Unterschreitet die durchschnittliche Seitenfehlerrate den Grenzwert E, so wird die Multiprogramming Level (Anzahl der Prozesse im Multiprogramming Set) wieder erhöht.

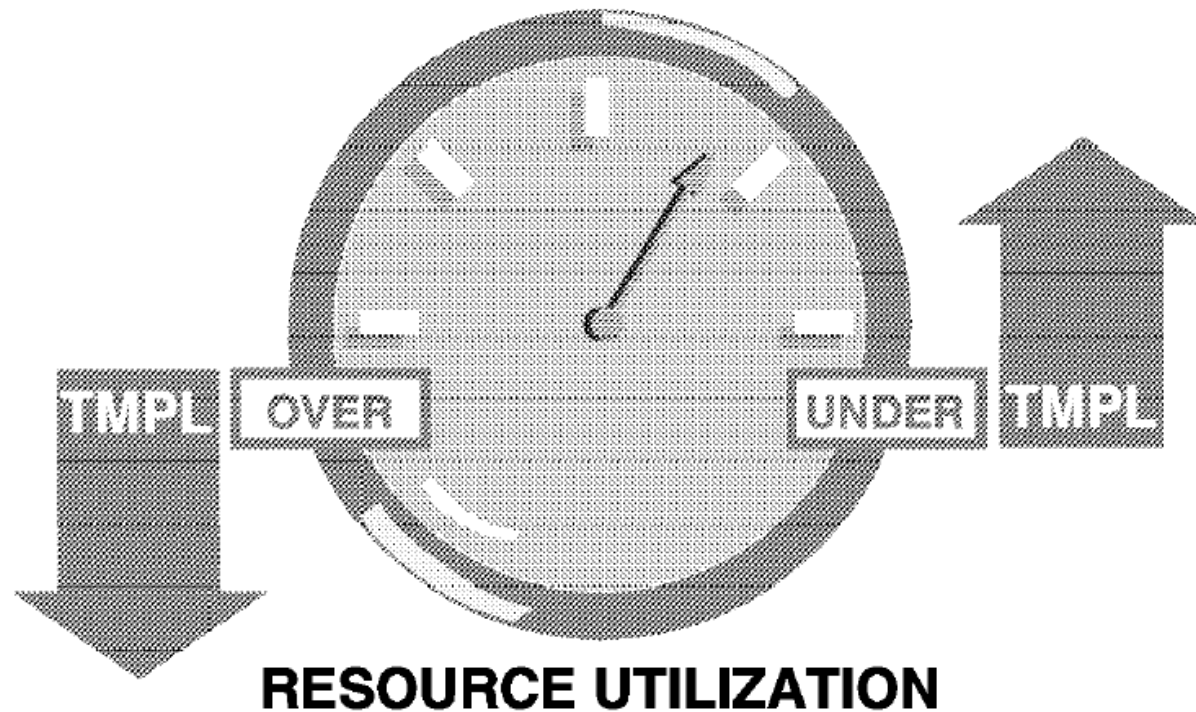
Wirkungsweise des Seitenwechsel-Begrenzers

Aktive und inaktive Prozesse

Aktive Prozesse befinden sich in jedem Augenblick in einem der drei Zustände laufend, wartend oder ausführbar. Inaktive Prozesse entstehen, wenn die Multiprogramming Level reduziert wird. Ihr Hauptspeichereinhalt wird auf einen externen Seitenspeicher (auxiliary storage) mit einem als Swap-out bezeichneten Vorgang ausgelagert.

Wird die Multiprogramming Level wieder erhöht, werden einige inaktive Prozesse wieder in den aktiven Zustand versetzt. Mittels eines als Swap-in bezeichneten Vorgangs wird ihr Hauptspeichereinhalt wieder hergestellt.

Der Work Load Manager benutzt komplexe Optimierungsalgorithmen, um unter Prioritätsgesichtspunkten zu entscheiden, welche Prozesse von einem Swap-out oder Swap-in betroffen sind.

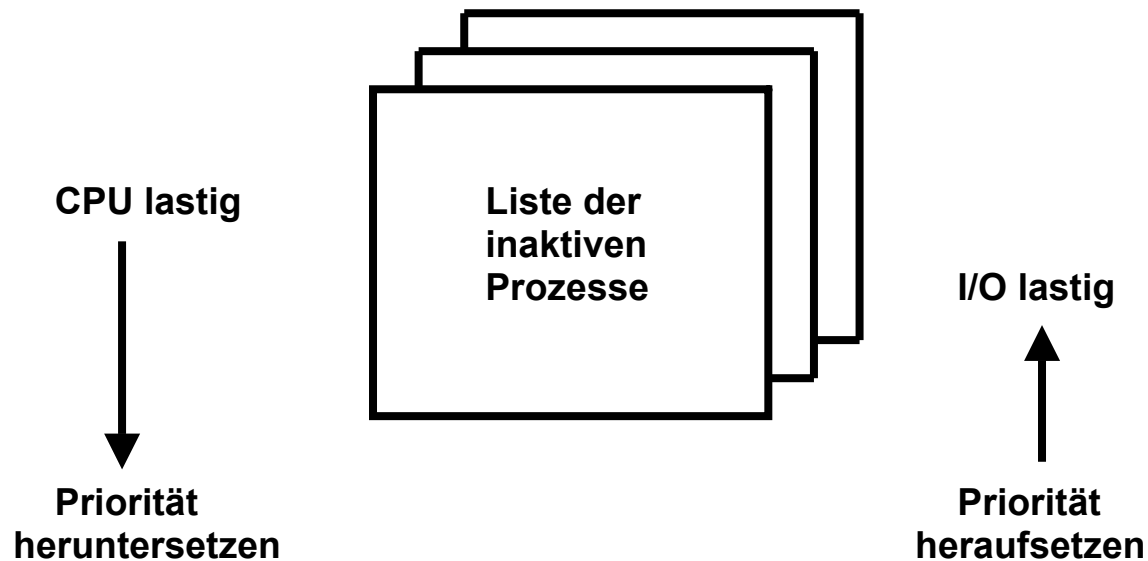


Als **Target Multiprogramming Level** (TMPL) bezeichnet man die Anzahl der swapped-in Address Spaces, die nach der Meinung von WLM das System mit seinen verfügbaren Ressourcen optimal bedienen kann.

Dieser Wert ändert sich ständig..

Die **Current Multiprogramming Level** (CMPL) ist die tatsächliche Anzahl der swapped in Address Spaces. Diese befinden sich untereinander im Wettbewerb um die verfügbaren System Ressourcen. CPML ist identisch mit der Anzahl der aktiven Prozesse.

WLM bemüht sich, den Unterschied zwischen CMPL und TMPL möglichst klein zu halten.



Prioritätssteuerung für inaktive Prozesse

WLM entscheidet unter Prioritätsgesichtspunkten, welcher inaktive Prozess für ein Swap-in selektiert wird. Die zugeordneten Prioritäten können sich ständig ändern. Ist z.B. zu einem Zeitpunkt die CPU Auslastung hoch, I/O Kapazität ist aber verfügbar, dann wird WLM die Priorität von CPU-lastigen Prozessen heruntersetzen, und die Priorität von I/O lastigen Prozessen heraufsetzen.

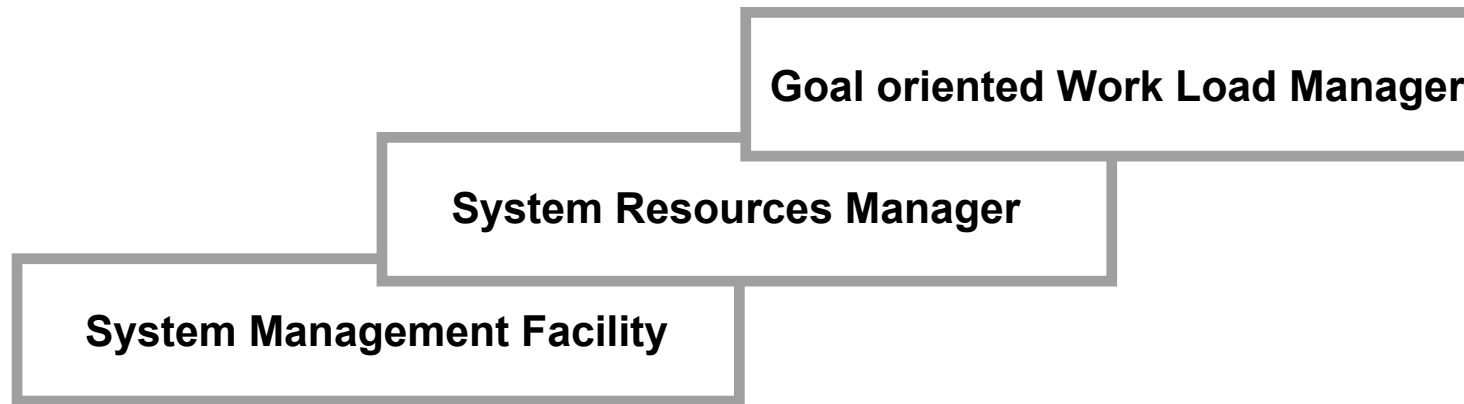
Automatische Steuerung der System-Ressourcen

Die unterschiedlichen Arten von Prozessen haben unterschiedliche Zielsetzungen. So sind zb. für viele interaktive CICS oder Webserver Zugriffe optimal Antwortzeiten (gemessen in Sekundenbruchteilen) auf Kosten einer optimalen Ressourcen Ausnutzung erwünscht. Bei Stapelverarbeitungsprozessen ist es umgekehrt. Es kann aber auch sein, dass für bestimmte Stapelverarbeitungsprozesse eine optimale Durchlaufzeit erwünscht ist.

Diese Zielsetzungen müssen dem System Resource Manager (SRM) vom Systemadministrator in irgend einer Form mitgeteilt werden. Die Zusammenhänge zwischen den unterschiedlichen Zielsetzungen sind jedoch sehr komplex, und ändern sich im Laufe eines Tages oder Monats ständig. Mit der zunehmenden Komplexität wird es für den Systemadministrator immer schwieriger zu entscheiden, welche von zahlreichen Einstellparametern er für einen optimalen Betrieb ändern muss. Oft kann er auch garnicht mehr schnell genug reagieren. Immer häufiger tritt das Problem auf, dass die Änderung von Systemparametern unerwartete und unerwünschte Seiteneffekte erzeugt. Bei der Vielzahl der Einstellungsmöglichkeiten ist es denkbar, dass eine Änderung keine Verbesserung, sondern eine Verschlechterung bringt.

Als Lösungsansatz wird dem Rechner die automatische Verwaltung und Steuerung aller System Ressourcen übertragen. Hierzu muss ihm aber gesagt werden, unter welchen Gesichtspunkten die Optimierung erfolgen soll.

Dies ist die Aufgabe des Goal Oriented Work Load Managers.

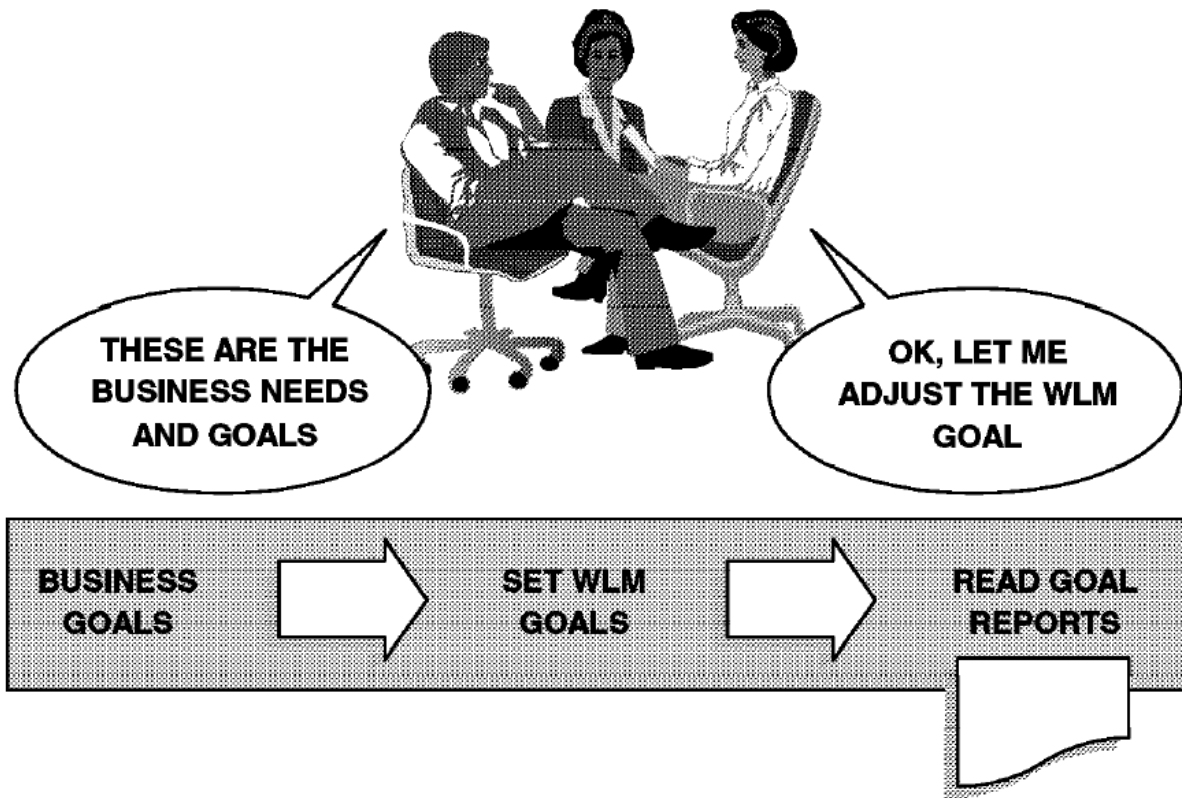


Vor der Einführung des Goal orientierte Workload-Managers existierte bereits der System Resource Manager (SRM). SRM war bereits in der Lage, die Verteilung der Betriebsmittel abhängig von der anfallenden Systemlast zu steuern. Die Verteilung der Betriebsmittel erfolgte abhängig von der anfallenden Systemlast.

Hierzu sammelt die System Management Facility (SMF) System-relevante Information über die Auslastung der einzelnen Systemkomponenten. SMF misst die gesamte Systemleistung und beobachtet die Nutzung der System Ressourcen (z.B. CPU, Hauptspeicher, Ein/Ausgabe).

Engpässe an Ressourcen können von SRM zum Beispiel durch eine Verringerung des Multiprogramming Sets und durch das Swapping von Adressenräumen aus dem Hauptspeicher in den externen Seitenspeicher aufgelöst werden.

Die Steuerung erfolgte durch die mehr oder weniger statische Eingabe von SRM Konfigurationsparametern durch den Systemadministrator, die sich nur in wesentlich geringem Umfang dynamisch unterschiedlichen Gegebenheiten anpassen ließen. Weiterhin ist es schwierig, bestimmte angestrebte Ziele (Business Goals, z.B. alle Transaktionen vom Typ x sollen eine Antwortzeit < 03 s haben) in SRM Konfigurationsparameter (z.B. Target Multiprogramming Level $\leq y$) zu übersetzen.



Der Goal oriented Work Load Manager

- reduziert den Aufwand für den System Administration
- reduziert die Notwendigkeit für Detailwissen

Goal Orientierung

Der Goal oriented Workload Manager (WLM) ist ein integraler Bestandteil des z/OS-Betriebssystems. Er erweitert den Funktionsumfang des SRM, und dehnt ihn zusätzlich auf den ganzen Sysplex aus.

Es ist die Aufgabe des Goal oriented Workload Managers, von Menschen gewünschte Optimierungs-Anweisungen in der Form von Geschäftsbegriffen (business terms) entgegen zu nehmen. Der Goal oriented Workload Manager übersetzt diese Anweisungen in System Resource Manager (SRM) Parameter.

Das Leistungsverhalten der Installation wird durch Vorgaben für Zielsetzungen (Business Goals) festgelegt. Der Work Load Manager setzt die Vorgaben in Einstellungen für den SRM um.

Instrumentierung

z/OS Subsysteme brauchen zusätzliche Funktionen, um mit WLM kooperieren zu können.

Der Workload-Manager unterstützt beispielsweise folgende Subsysteme :

- IMS
- JES2/JES3
- TSO/E
- CICS
- OMVS
- DB2
- TCP
- SNA
- Web Server
- WebSphere
- MQSeries
- non-z/OS partitions,
z.B zLinux

Die Integration der Subsysteme ist ein Alleinstellungsmerkmal des z/OS Work Load Managers. Manche Subsysteme unterhalten ihre eigenen Work Load Managent Komponenten. Beispiele sind CICS und WebSphere. CICS Transaktionen können nach Ihrem Namen (TRID) klassifiziert werden. Die z/OS WebSphere Version unterscheidet sich von den WebSphere Versionen auf anderen Betriebssystem Plattformen dadurch, dass alle z/OS WLM Funktionen in Anspruch genommen werden können.

Work Unit (Arbeitseinheit)

Die von einem Rechner zu verarbeitende Workload besteht aus vielen einzelnen Work Units (Arbeitseinheiten). Eine Work Unit oder Arbeitseinheit ist eine Verarbeitungsaufgabe für einen Rechner. Beispiele für Work Units sind

- CICS Transaktionen,
- TSO Sitzungen,
- Batch Jobs.

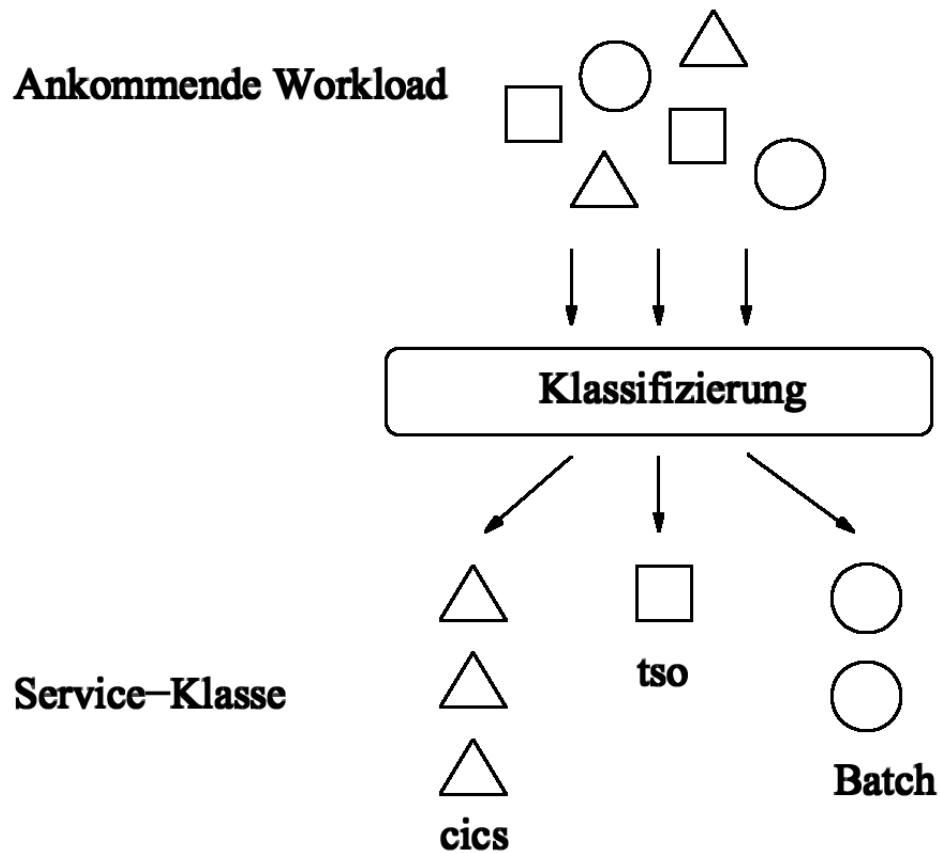
In einem z/OS Rechner werden in jedem Augenblick zahlreiche Work Units parallel verarbeitet.

Der Benutzer definiert mittels Zielvorgaben, welche Leistungen unter welchen Umständen von dem Rechnersystem zu erbringen sind. Das System überwacht während der Abarbeitung die Workloads, vergleicht ihre Laufzeitergebnisse mit den Zielvorgaben (Business Goals) und passt den Zugang zu und Verbrauch an Ressourcen dynamisch auf der Basis der Vorgaben (Goals) an.

Theoretisch wäre es denkbar, für jede einzelne Work Unit eigene Zielvorgaben zu erstellen. Da dies unpraktisch ist, fasst man Work Units mit ähnlichen Zielvorgaben zu „Service Klassen“ zusammen. Alle in einer Service Klasse zusammengefassten Work Units erhalten die gleichen Zielvorgaben.

Die Anzahl der Service Klassen sollte nicht zu niedrig (schlechte Optimierung) und nicht zu hoch (hoher Administrationsaufwand und WLM Verarbeitungsaufwand) sein.

25 Service Klassen in einer Installation ist eine gute Faustformel.



Die Workload für einen Rechner besteht aus vielen unterschiedlichen Arbeitseinheiten (Work Units).

Zur Definition der Ziele (Goals) werden alle auftretenden Verarbeitungsanforderungen (Work Units) in „Service-Klassen“ eingeteilt. Eine mögliche (sehr einfache Aufteilung könnte z.B. drei Service-Klassen vorsehen:

- CICS Jobs,
- TSO Jobs,
- Batch Jobs

Es ist aber auch möglich, z.B. CICS Jobs entsprechend ihrer User ID oder ihrer TRID in mehrere unterschiedliche Service-Klassen aufzuteilen

Service Klassen

Service-Klassen (Dienst Klassen) sind Einheiten von Workload mit sehr ähnlichen Charakteristiken, für die die Zielvorgaben definiert werden. Die Service-Klasse stellt das Grundkonstrukt im *Goal Mode* dar. Sie ist das Ergebnis der Klassifizierung der Workloads mit unterschiedlichen Leistungsmerkmalen in Gruppen, die mit den gleichen Zielvorgaben versehen werden können.

Einordnung in Dienstklassen

Classification Rules werden benutzt, um die Menge aller möglichen Arbeitsanforderungen in Dienstklassen (Service Classes) einordnen (klassifizieren). Die Klassifikation basiert auf den Attributen einer individuellen Arbeitsanforderung. Dies kann sein:

- User ID
- Art des aufgerufenen Prozesses (Transaktionstyp, Stapel,)
- die Arbeits-Umgebung oder das Subsystem
- Accounting Information
-

Jeder Dienstklasse sind „Ziele“ zugeordnet

- Antwortzeit (Response Time)
- Geschwindigkeit (Velocity)
- Stellenwert (Importance)
- andere (discretionary)

Jede Dienstklasse besteht aus zeitlichen Perioden (z.B. 10 Sekunden):

- Während einer Periode sind begrenzte Ressourcen verfügbar (z.B. CPU Zyklen, I/O Zugriffe,
- Nach Ablauf der Periode i erfolgt eine Migration nach Periode $i + 1$
- Für jede Periode können unterschiedliche Ziele existieren.

Eine realistische Annahme ist, dass nicht alle Ziele erreicht werden können. WLM plaziert Arbeitsanforderungen so, daß die Wahrscheinlichkeit, alle Ziele zu erreichen, optimiert wird.

Service Definition

Eine Workload-Manager Service Definition wird mittels einer speziellen administrativen Anwendung, der *WLM Administrative Application*, erstellt und abgeändert. Diese kann unter der *Interactive System Productivity Facility* (ISPF) von autorisierten TSO-Benutzern gestartet werden.

Der wichtigste Punkt bei der Definition einer Service-Klasse ist es festzulegen, wie wichtig sie ist, um die übergeordneten Geschäftsziele zu erreichen. Bei einem optimierten System kann es durchaus sein, dass nicht alle Ziele erfüllt werden können.

Die Wichtigkeit wird dargestellt, indem eine *Business Importance* für die Service-Klasse definiert wird. Diese *Business Importance* ist später für das System der wesentliche Entscheidungsfaktor, wenn der Zugang zu den Ressourcen geregelt werden muss.

Service Units

System Ressourcen werden von Prozessen verbraucht. Für WLM werden Prozesse durch die Adressenräume (Address Spaces) charakterisiert, in denen sie ablaufen. Im Falle von CICS sind dies die als Enclaves bezeichneten Thread-ähnlichen Einheiten.

Die von den Prozessen verbrauchten Ressourcen werden von SRM in „Service Units“ gemessen. Eine Service Unit ist eine Meßgröße, in die Parameter wie

- verbrauchte CPU Zeit
- I/O Aktivität
- Hauptspeichernutzung

eingehen.

Work Load Manager Operation

Die System Management Facility (SMF) Komponente des Workload-Managers sammelt Daten über den aktuellen Zustand aller durch WLM verwalteten Ressourcen. Dies sind Informationen über die Prozessoren, den Hauptspeicher, die Benutzung der I/O Einheiten und die von WLM für die von Anwendungen verwaltete Warteschlangen. Das Sampling Intervall beträgt z.B. 250 Millisekunden. Zeiträume, in denen die Work Unit nicht gearbeitet hat und auch keine Anforderungen an das System gestellt hat, werden nicht berücksichtigt. Dasselbe gilt für Zustände, die vom Workload-Manager nicht erfasst werden können, wie zum Beispiel das Warten auf die Freigabe eines Locks, das durch eine Anwendung verwaltet wird.

Aus den gesammelten Daten wird für jede Arbeitseinheit (Work Unit) festgestellt, ob sie Ressourcen benutzt (**Using**) oder darauf wartet (**Delay**). Der Workload-Manager verwendet diese Daten, um die Verteilung von Ressourcen auf Service Classes vorzunehmen. Da sie die Basis für die WLM-Algorithmen darstellen, können sie auch für die Definition von Zielen verwendet werden.

Für die Definition von Zielvorgaben bietet der z/OS-Workload-Manager drei Arten von Zielen:

- Response Time Goals
- Execution Velocity Goals
- Discretionary

Arten von Zielen

Response Time Goals

Unter einem Response Time Goal versteht man die Zeitspanne zwischen Start und Fertigstellung einer Work Unit, die in Sekunden pro Programm gemessen wird. Als Zeit wird die vollständige Verweildauer inklusive der Zeit, in der die Work Unit nicht arbeitet, betrachtet. Dadurch kann man die Wartezeit eines Benutzers erkennen.

Execution Velocity Goals

Wenn zu wenige Work Units in einer Service-Klasse enden, kann ein Response Time Goal nicht verwendet werden. Um für derartige Work Units Ziele definieren zu können, kann ein Execution Velocity Goal verwendet werden. Execution Velocity legt den Anteil der akzeptablen Wartezeit bei der Ausführung von Programmen fest. Bei einem Execution Velocity Goal werden nur die Zustände betrachtet, bei denen eine Work Unit Ressourcen benutzt (**Using**) oder darauf wartet (**Delay**).

Discretionary

Für eine Gruppe von Work Units existieren keine bestimmten Zielvorgaben. In z/OS Systemen werden diese Gruppen von Work Units als *Discretionary* bezeichnet. Diese Work Units erhalten nur dann Ressourcen, wenn diese ausreichend vorhanden sind, und sie sind die Ersten, die keinen Zugang mehr erhalten, wenn es eng im System wird. Auf der anderen Seite tritt ein Workload Management Regelmechanismus in Kraft, der den Ressourcen Zugang für Serviceklassen mit Zielvorgaben begrenzt, wenn diese ihre Ziele deutlich übererfüllen.

Response Time

Response Time oder Antwortzeit drückt den Wunsch aus, dass die Zeit, die Transaktionen im System verweilen, maximal einem vorgegebenen Wert entsprechen. Es gibt 2 alternative Möglichkeiten, dieses Ziel zu spezifizieren

Durchschnittliche Antwortzeit (Average Response Time). Beispiel: Avg. Resp. Time = 0,75 s

Prozentsatz Antwortzeit (Percentile Response Time). Beispiel % Resp. Time: 90 % < 0,5 s (90 % aller Transaktionen werden in weniger als 0,5 s beantwortet).

Die Percentile Response Time ist häufig wichtiger als die Average Response Time (wie häufig ist die Antwortzeit zu lang ?).

Die Antwortzeit betrifft nur die Zeit zwischen Eintreffen und Verlassen der Nachricht in System. Netzverzögerungen werden von WLM nicht erfasst und deshalb auch nicht berücksichtigt.

Execution Velocity

Bei einem *Execution Velocity Goal* werden nur die Zustände betrachtet, bei denen eine Work Unit Ressourcen benutzt (*using*) oder darauf wartet (*delay*). Ein ausführbarer Prozess (nicht laufend) verbraucht keine Ressourcen.

Velocity = theoretisch beste Zeit / wirklich verbrauchte Zeit

Execution Velocity ist wie folgt definiert:

$$\text{Execution Velocity} = \frac{\text{Total Usings}}{\text{Total Usings} + \text{Total Delays}} \times 100$$

Näherungsweise ist dies:

$$\text{Velocity} = \frac{\text{Zeit für CPU} + \text{I/O}}{\text{Zeit für CPU} + \text{I/O} + \text{paging} + \text{MPL} + \text{swpin} + \text{queues}} \quad \text{MPL} = \text{Multiprogramming Level}$$

Nehmen wir z.B. an, dass eine Work Unit fünf Mal auf Ressourcen wartet und fünf Mal Ressourcen benutzt. Man erhält man eine *Execution Velocity* von 50.

$$\text{Execution Velocity} = \frac{5 \text{ Usings}}{5 \text{ Usings} + 5 \text{ Delays}} \times 100 = 50$$

Der Wert der Execution Velocity liegt immer zwischen 1 und 100.

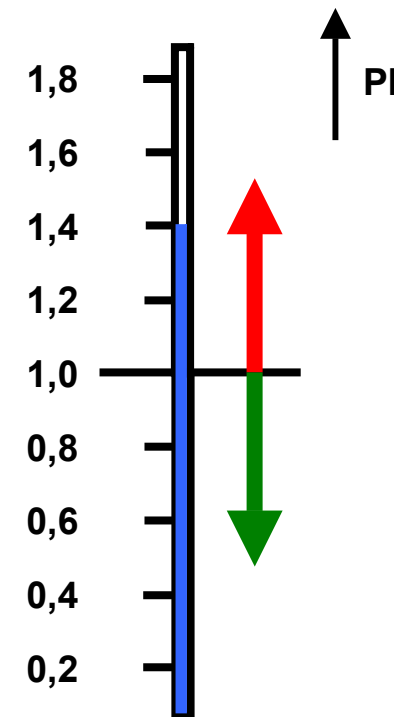
Performance Index

Frage: Wie erfüllen Service-Klasse ihre Ziele und wie verhalten sie sich im Vergleich zu anderen Service-Klassen.

Workload-Manager-Algorithmen lösen die dieses Problem durch die Definition eines **Performance Index (PI).**

Die Definition des Performance Index (PI)besagt:

- PI < 1:** Ziel wurde übererfüllt. Die Service Class benötigt vermutlich nicht alle ihr zugeteilten Ressourcen
- PI = 1:** Ziel wurde exakt erreicht. Alles ist ok
- PI > 1:** Ziel wurde nicht erreicht. Die Service Class benötigt zusätzliche Ressourcen



Performance Index Definitionen

Die Definition des Performance Index (PI) erfolgt unterschiedlich für die Execution Velocity Goals und die Response Time Goals, um die gezeigte einfache Darstellung zu ermöglichen.

Definition des Performance Index für das Execution Velocity Goal:

$$\text{PI} = \frac{\text{Execution Velocity Goal}}{\text{Aktuell erreichte Execution Velocity}}$$

Definition des Performance Index für das Response Time Goal:

$$\text{PI} = \frac{\text{Aktuelle Response Time}}{\text{Response Time Goal}}$$

Ein $\text{PI} < 1$ bedeutet, Ziele wurden erreicht

Ein $\text{PI} > 1$ bedeutet, Ziele wurden nicht erreicht

Aussagen des Performance Index

Der Performance Index für Response Time Goals und Execution Velocity Goals kann aus den Zieldefinitionen und den aktuell gemessenen Werten ermittelt werden. In beiden Fällen bedeutet ein Wert von > 1 , dass das Ziel nicht erfüllt wird und dass die Workload-Manager-Algorithmen aktiv werden müssen, um den Engpass zu beheben.

Ein Wert von 1 signalisiert Zielerfüllung. Ist ein Wert < 1 , dann ist das Ziel übererfüllt.

Performance Index

Wichtigkeit (Importance)

In einem gut ausgelasteten System werden nicht notwendigerweise alle Ziele erreicht.

Neben der Zielvorgabe wird jeder Service Class (Dienstklasse) eine Wichtigkeit (Importance) zugeordnet, die festlegt, welche Klassen bevorzugt bzw. benachteiligt werden sollen, wenn die Ressourcen im System nicht mehr ausreichend zur Verfügung stehen.

Der wichtigste Punkt bei der Definition einer Service-Klasse ist es festzulegen, wie wichtig sie ist, um die übergeordneten Geschäftsziele zu erreichen. Die Wichtigkeit wird dargestellt, indem eine *Goal Importance* für die Service-Klasse definiert wird. Diese *Goal Importance* ist später für das System der wesentliche Entscheidungsfaktor, wenn der Zugang zu den Ressourcen geregelt werden muss. Sie wird normalerweise durch eine Ziffer zwischen 1 und 5 angegeben, wobei 1 die höchste, und 5 die niedrigste Wichtigkeit darstellt.

Dies wird an Hand eines Beispiels erläutert. Die gesamte Workload für den Rechner wird in 5 Dienstklassen aufgeteilt. Hierbei werden die Stapelverarbeitungs-Work Units in zwei Gruppen mit unterschiedlichen Zielsetzungen aufgeteilt.

CICS Work Units

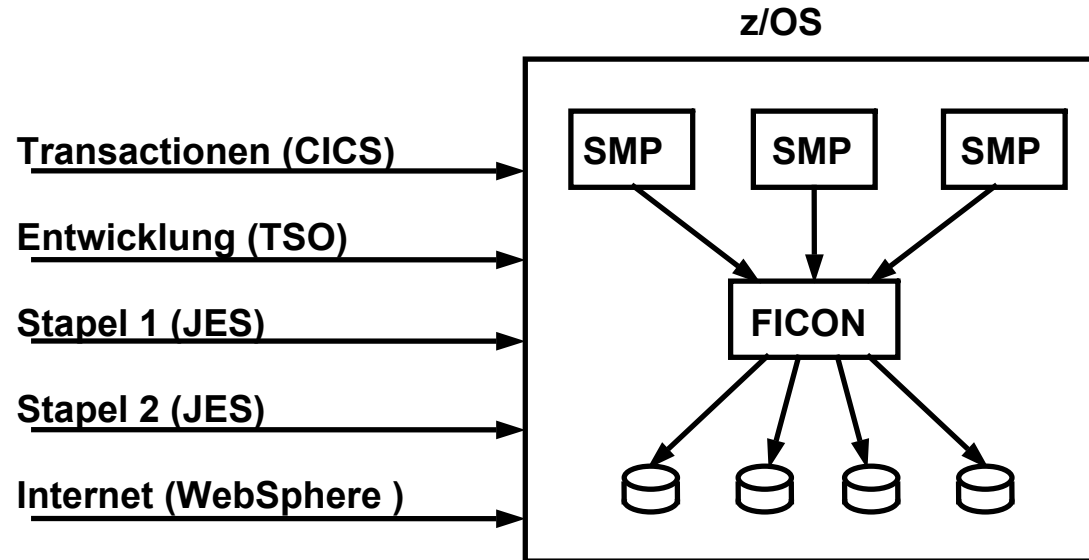
TSO Work Units

Stapel 1

Stapel 2

Internet

Die folgende Seite zeigt ein Beispiel für die Classification Rules.



Classification Rules Beispiel

Unterschiedliche Ziele für Service Classes:

Wichtigkeit

Transaktionen, (CICS)	90 % Antwortzeit < 0,35 s	1
Stapel 1,	90 % complete < 3 Stunden	4
Stapel 2,	niedrige Priorität	5
Internet (DB2) *),	90 % Antwortzeit < 0,3 s	3
Entwicklung (TSO),	90 % Antwortzeit < 0,57 s	5

*) DB2 Stored Procedures werden in WLM-managed Adressenräumen ausgeführt.

WLM Regelmechanismus

WLM benutzt einen Regelmechanismus, um zur Laufzeit den Zugang zu den Ressourcen zu steuern. Dazu werden kontinuierlich Daten aus dem z/OS-System gesammelt (mit Hilfe der System Management Facility, SMF). Dies sind Informationen über die Wartezustände der Arbeitseinheiten auf die Ressourcen, die Anzahl der laufenden Arbeitseinheiten und deren Abarbeitungszeiten. Die Informationen werden in Service Classes (Dienstklassen) zusammengefasst entsprechend der Klassifizierung, die durch den Systemverwalter vorgenommen wurde. Dann wird auf der Basis dieser Informationen die Zielerfüllung für jede Klasse berechnet und, falls notwendig, der Zugang zu den Ressourcen angepasst.

Die Anpassung erfolgt immer in Abhängigkeit von der Wichtigkeit (Importance) der Klassen, und dem Grad, in dem das Ziel verfehlt wird. Das heißt, die wichtigste Klasse, die am weitesten ihr vorgegebenes Ziel verfehlt hat, wird als erste betrachtet und die Klassen mit der geringsten Wichtigkeit sind die potenziellen Kandidaten, um Ressourcen abzugeben. Die wichtigere Klasse wird der Empfänger (**Receiver**) zusätzlicher System Ressourcen. Diese müssen natürlich einer anderen Dienstklasse weggenommen werden. Dabei wird allerdings berücksichtigt, ob ein potenzieller Spender (**Donor**) auch tatsächlich das benötigte Ressourcen verwendet.

Hieraus werden vom Goal Oriented Work Load Manager Einstellparameter für den System Resources Manager (SRM) abgeleitet. Der SRM nimmt dann die entsprechenden Anpassungen vor.

Dieser Regelmechanismus läuft typischerweise alle 10 Sekunden im z/OS ab. In der Zwischenzeit werden die Daten für das nächste Berechnungsintervall von SRM gesammelt. Ein Berechnungsintervall endet, wenn eine Anpassung zugunsten einer Dienstklasse durchgeführt werden kann.

Alle 250 ms sammelt und misst SMF Daten



Alle 10 s Zusammenfassung der Daten, Erfüllung der Ziele evaluieren



Wähle Receiver aus



Ermittle Receiver Engpass



Behebe Receiver Engpass
Ermittle Donor
Schätze Auswirkung der Änderung ab
Führe die Anpassung durch



Existieren weitere Engpässe



Ende

WLM Regelmechanismus

Der in jedem Zeitintervall ablaufende Regelmechanismus besteht aus den folgenden Schritten:

- Auslastung messen
- Verhalten mit den Zielen (Goals) vergleichen
- Anpassungen vornehmen (Donor und Receiver ermitteln).

WLM-managed Initiators

Für JES2 entscheidet der WLM (Workload Manager) , wann wieviele Initiators auf welchen Systemen des Parallel Sysplex-Verbundes gestartet und gestoppt werden. Der WLM stützt sich dabei auf die Vorgaben der Performance Goals in der WLM-Policy.

Simulationsmodell

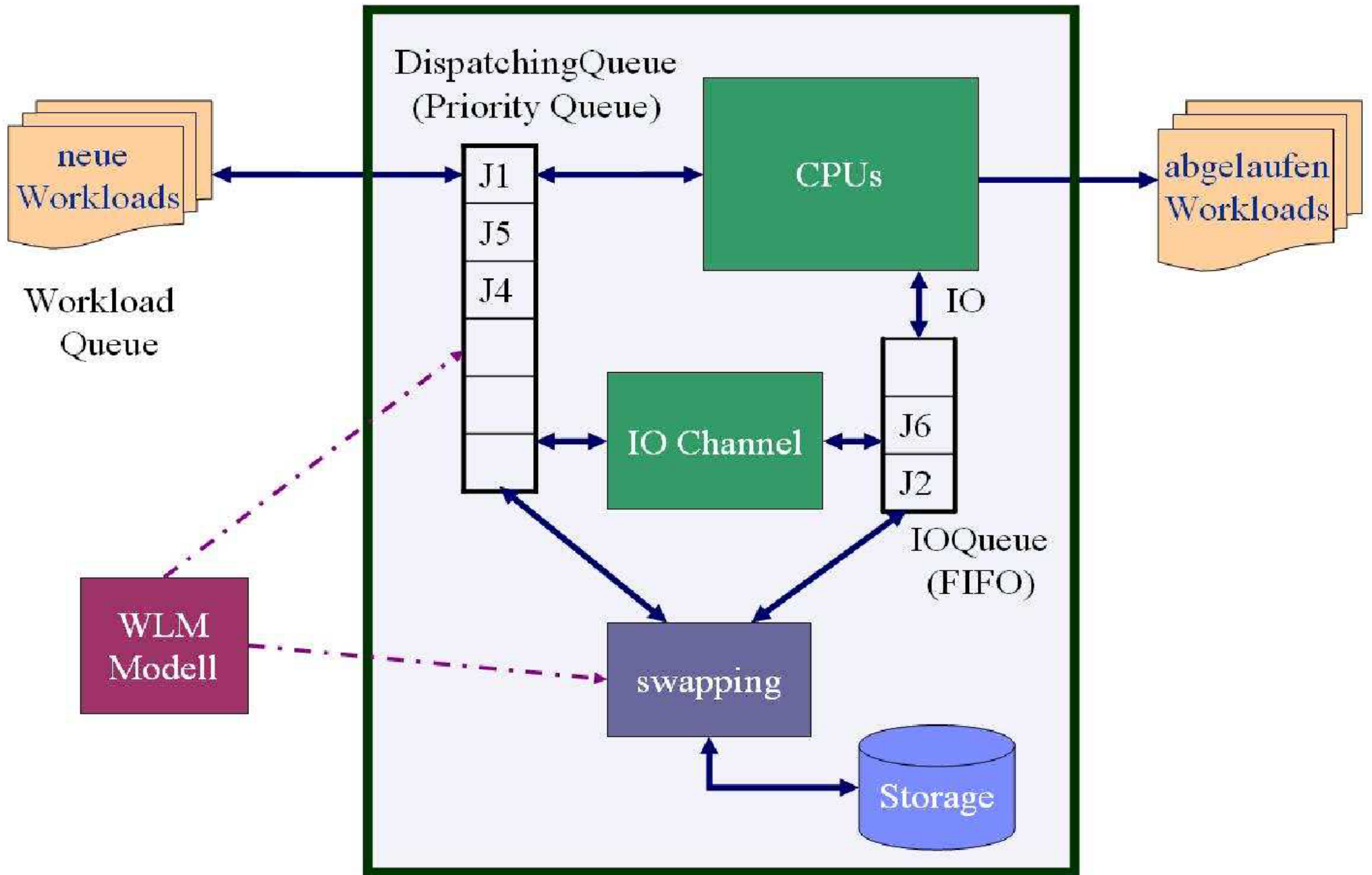
Wir beobachten diese Vorgänge an Hand eines einfachen Simulationsmodells, welcher in der folgenden Abbildung dargestellt ist.

Das simulierte System besteht aus mehreren CPUs, welche gleichzeitig ausführbare Prozesse bearbeiten. Die wartenden Prozesse befinden sich in der IOQueue. Nach Ende der I/O Verarbeitung werden die jetzt ausführbaren Prozesse der Dispatching Queue übergeben.

Der Zugang zu den CPUs wird durch die Verwendung der Dispatching-Queue gesteuert. Jeder Prozess, d.h. jede im Hauptspeicher befindliche Work Unit, erhält eine Dispatch-Priorität und ist innerhalb dieser Dispatch-Queue nach seiner Priorität geordnet. Der Prozessscheduler ordnet immer dem ersten Prozess in der Dispatch-Queue mit der höchsten Priorität einer verfügbaren CPU zu.

Unser Modell enthält weiterhin einen Ein-/Ausgabe Kanal (I/O-Channel). In z/OS ist dieser Betriebsmittelzugang durch die Festlegung von Prioritäten für den Ein-/Ausgabe-Kanal geregelt. Um das in der Simulation verwendete Szenario einfach und anschaulich zu halten, wurde statt dessen eine FIFO-Queue (First In First Out) als Ein/Ausgabe-Queue (I/O-Queue) verwendet. Hierbei werden alle Prozesse in der Reihenfolge ihres Eingangs bearbeitet.

Für die Speicherverwaltung ist ein Swapping modelliert. Wir nehmen an, dass sich der Prozess, der in der Dispatching-Queue oder Ein/Ausgabe-Queue liegt, im Hauptspeicher befindet. Wenn die Workload ungesteuert abläuft, kann es zu Engpässen bei den Betriebsmitteln kommen. Um das zu vermeiden, wird für jede Service-Klasse festgelegt, wie viele Adressräume gleichzeitig im System sein dürfen und wie viele auf jeden Fall im System bleiben sollen. Die Zahl der Adressräume, die gleichzeitig im System sind, wird in der Literatur als Multiprogramming Level (MPL) bezeichnet. Der MPL ist Teil eines wesentlichen Steuerungsmechanismus des Systems. Die Unter- und Obergrenze für den MPL darf nur durch den WLM dynamisch geändert werden. Wenn das System feststellt, dass Engpässe auftreten, kann der Workload-Manager den MPL der Service-Klasse reduzieren, damit Adressräume ausgelagert werden und die verbleibenden Adressräume effizienter arbeiten können.



Service-Klassen Definition für das Experiment

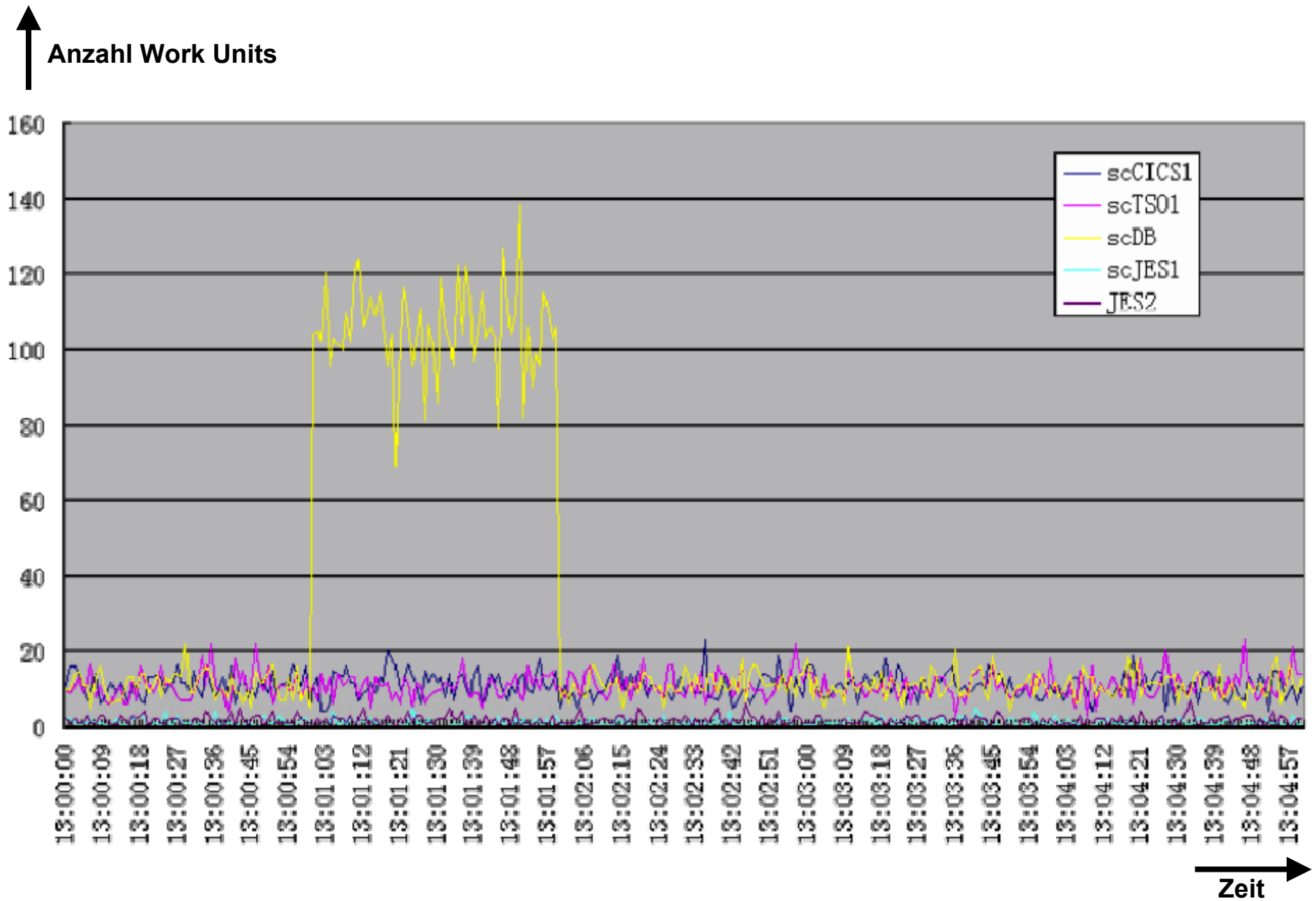
Für das Simulationsmodell werden insgesamt 5 Service-Klassen modelliert:

Name	Importance	Goal
scCICS1	1 (hoch)	0,35 s Response
scTSO1	2	0,57 s Response
scDB21	3	0,3 s Response
scJES1	4	15 Velocity
scJES2	5 (niedrig)	15 Velocity

In der folgenden Graphik ist die Verteilung der Work Units in den einzelnen Service-Klassen für die Zeit von 13:00:00 bis 13:05:00 wiedergegeben. Während der ganzen Zeit liegt die Anzahl Work Units in jeder Klasse unter 20. Mit einer Ausnahme:

Um 13:01:00 wird eine große Anzahl von Work Units in der Service Klasse ‚scDB21‘ gestartet. Der Wert steigt plötzlich von unter 20 auf über 100. Un 13:02:00 fällt der Wert wieder auf unter 20 zurück.

Wir sehen uns an, wie WLM den PI Index managed.



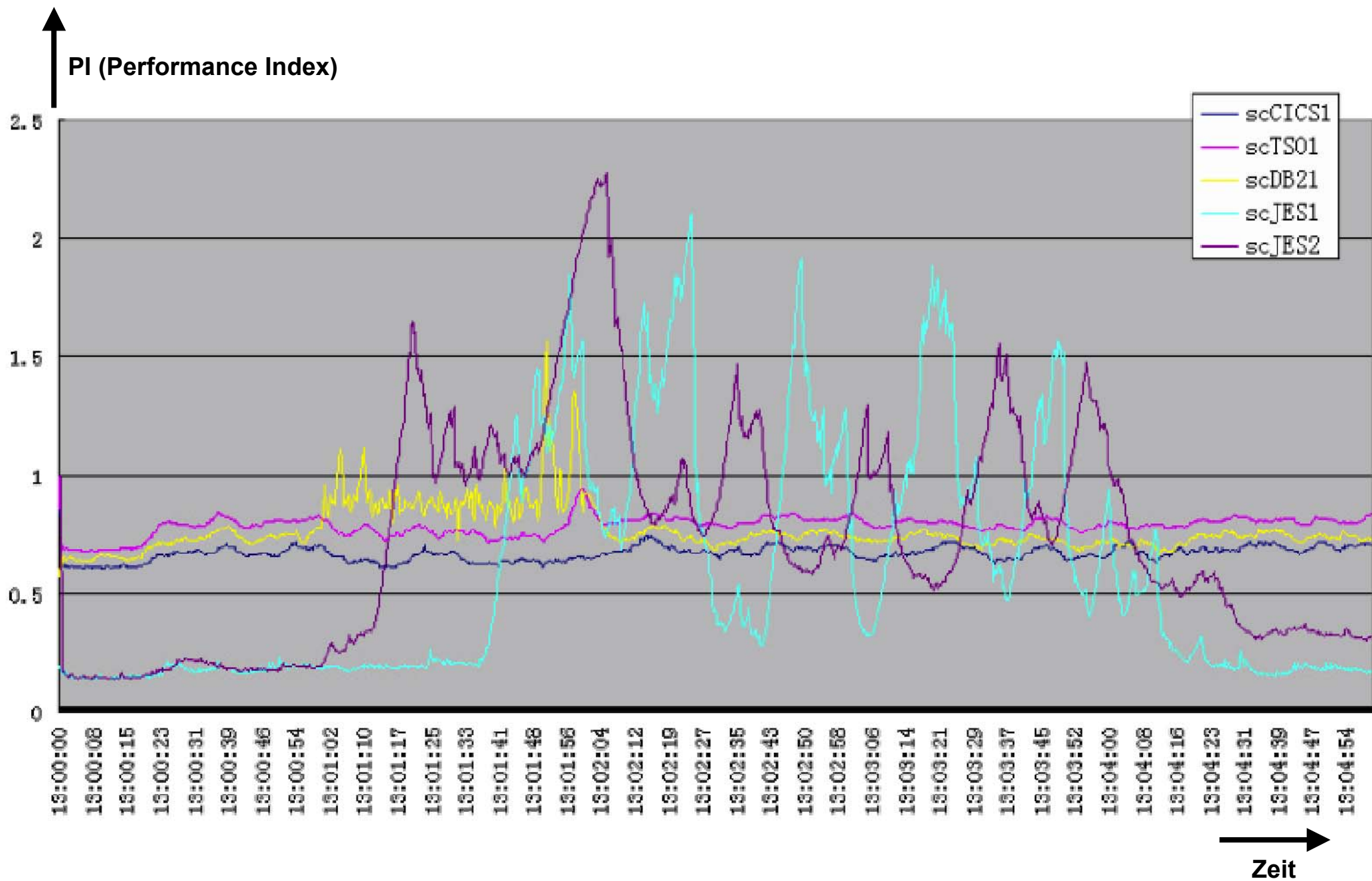
WLM arbeitet typischerweise mit 10 Sekunden Intervallen. In dem hier gezeigten Beispiel sind es 9 Sekunden.

Reaktion des Work Load Managers

In der folgenden Abbildung ist der zeitliche Verlauf der PIs (Performance Indizes) für die 5 Service Klassen scCICS1, scTSO1, scDB21, scJES1 und scJES2 dargestellt. Da nur wenige Work Units bis zu 13:01:00 in das System eintreten, und die CPU-Anforderungen gering sind, werden die Ziele aller Service-Klassen übererfüllt (PI <1). Die Ziele für scJES1 und scJES2 sind nicht so eng gefasst, deshalb ist es auch möglich, alle Ziele deutlich über zu erfüllen. Da seit 13:01:02 die Anforderungen durch die zusätzlichen scDB21-Work Units sehr hoch sind, werden zu diesem Zeitpunkt die Ziele für scDB21 nicht mehr erreicht (PI >1). WLM beginnt jetzt zu steuern.

Um 13:01:02 war die Service Klasse scDB21 der Receiver zusätzlicher Ressourcen, die von dem Donor scJES2, der Service Class mit der niedrigsten Importance, abgegeben wurden. Wir sehen, dass etwa 10 Sekunden später (um 13:01:10) der PI für scDB21 wieder unterhalb von 1 ist und der PI für scJES2 deutlich oberhalb von 1 steigt, da die MPL-Werte der scJES2 Service Klasse reduziert werden.

Um 13:01:10 hatte die Performance der Service Klasse scJES1 einen PI deutlich kleiner als 1, während der PI Wert von scJES2 deutlich über 1 liegt. Obwohl die Importance von scJES1 höher als die von scJES2 ist, wird scJES1 zum Donor und gibt Ressourcen an scJES2 (Receiver) ab. Danach ist bis um 13:01:40 der PI für scJES1 gestiegen und der PI für scJES2 besser geworden, obwohl die Ziele von scJES2 immer noch nicht erfüllt werden (der PI ist > 1).



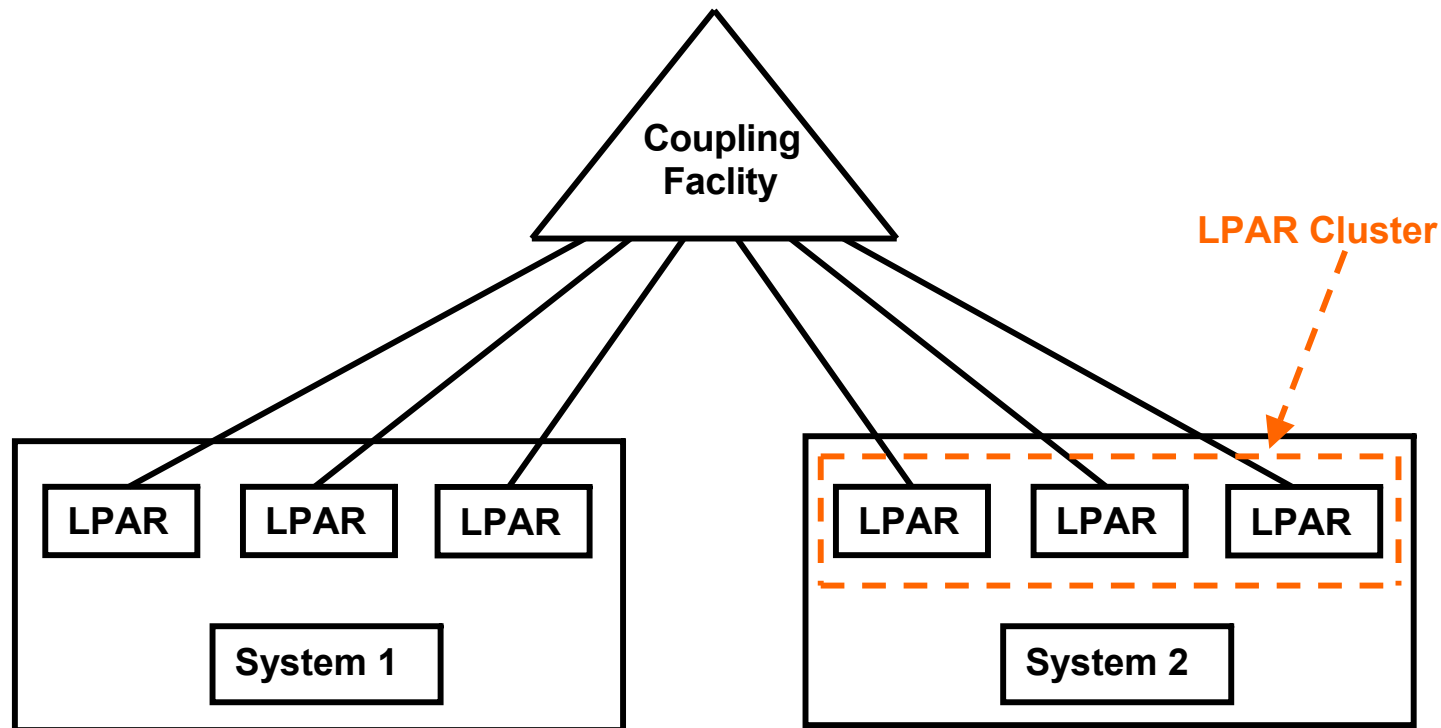
Um 13:01:41 gibt es zwei Service-Klasse scJES1 und scJES2 mit einen PI größer als 1. Der PI von scDB2 liegt unter 1. Service Klasse scDB2 gibt deshalb Ressourcen an Service Klasse scJES1 ab. Das Ergebnis ist wenig erfreulich, denn

um 13:01:48 haben die Service-Klassen scDB21, scJES1 und scJES2 alle ihre Ziele nicht erreicht. Die Service Klasse scDB2 wird deshalb ein Receiver zusätzlicher Ressourcen und die Service Klasse scTSO1 wird s der Donor. Die PI von scTSO1, steigt, bleibt aber unter dem Wert 1.

Ab 13:02:00 fällt die Anzahl von Work Units in der Service Klasse scDB21 wieder auf den ursprünglichen Wert < 20. Die PI Werte von scDB2 und scJES1 sind deshalb besser geworden. Da die PI der scJES2 immer noch größer als 1 ist, führt WLM

ab 13:02:04 bis 13:04:00 führt WLM weitere Anpassungen durch. Dies verursacht wilde Oszillationen zwischen dem PI von scJES1 und dem PI von scJES2. Um 13:05:00 endet das Test.

Der Work Load Manager des Simulationsmodells verwendet einen sehr einfachen Regelalgorithmus. Offensichtlich sind Verbesserungen im Regelalgorithmus notwendig, z.B. um die Oszillationen zwischen 13:02:04 und 13:04:00 zu beseitigen. Der Algorithmus des z/OS Work Load Managers ist dagegen ein sehr komplexes Gebilde, in das vieljährige Erfahrungen eingeflossen sind. Nach allgemeiner Meinung liefert WLM deshalb auch in komplexen Installationen sehr befriedigende Ergebnisse.

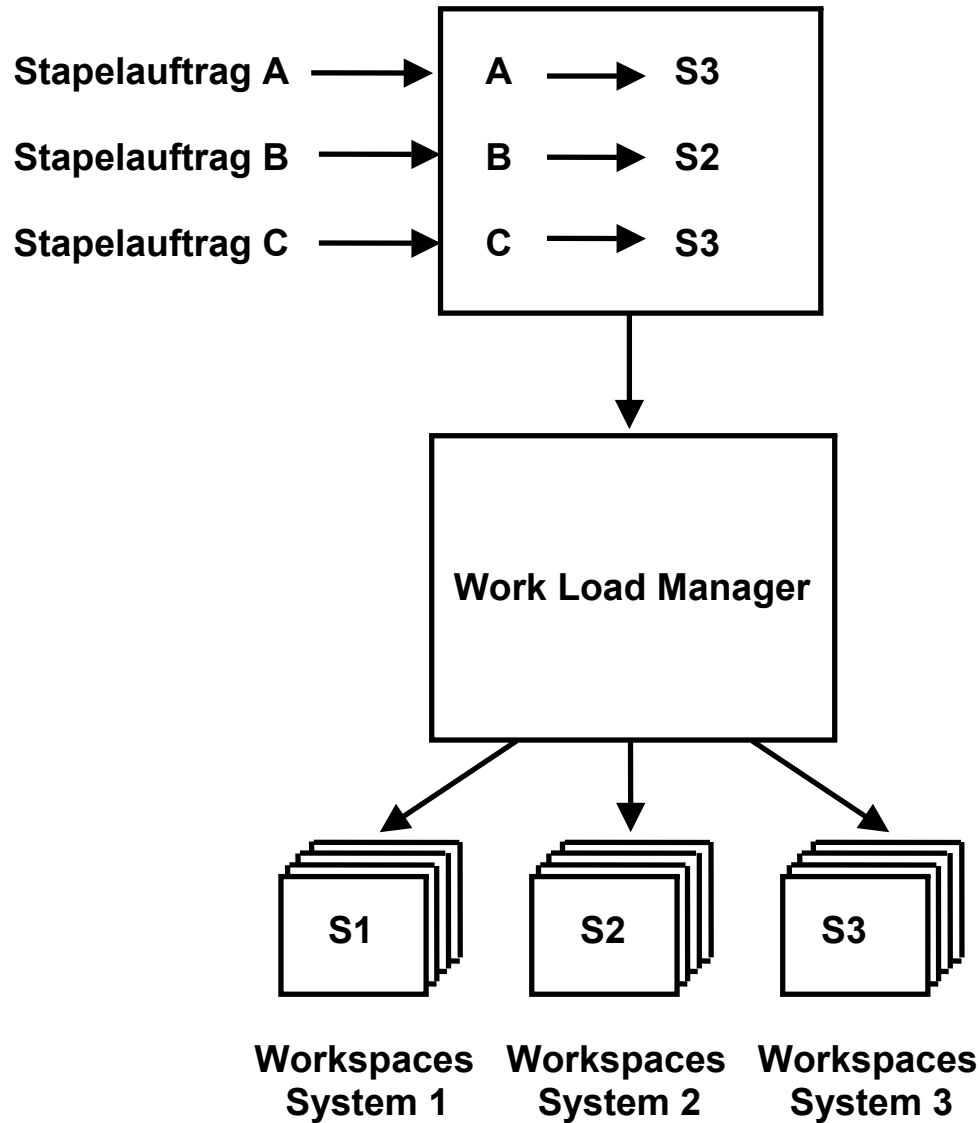


WLM und Sysplex

Eine große Sysplex Installation besteht in der Regel aus mehreren physischen Rechnern (Systeme in der IBM Terminologie), von denen jeder mehrere LPARs mit je einem SMP (z/OS Instanz) pro LPAR enthält. Jede z/OS Instanz in jeder LPAR verfügt über ihren eigenen WLM, und jede z/OS Instanz ist über ein Coupling Link mit der Coupling Facility verbunden.

Alle LPARs auf dem gleichen System werden als **LPAR Cluster** bezeichnet, und können von WLM unter Benutzung der IRD Funktionen gemanaged werden

Die WLM Instanzen auf unterschiedlichen Systemen tauschen Information über die Coupling Facility aus. Alle LPARs haben die gleiche Service Klassen Definitionen.



Eine Aufgabe des Work Load Managers ist es, Einstellungen der einzelnen Systeme eines Sysplex dynamisch an sich ändernde Belastungen anpassen und automatisch justieren.

Bei widersprüchlichen Anforderungen (Regelfall) verfügt der WLM über Algorithmen, einen möglichst optimalen Kompromiss zu erreichen.

Gezeigt ist, wie die gerade neu eintreffenden Jobs A, B und C auf die Systeme S1, S2 und S3 verteilt werden. Der Work Load Manager entscheidet (auf Grund seiner Einschätzung der derzeitigen Auslastung) Job B an System S2 sowie Jobs A und C an System S3 zu vergeben, während System S1 keine zusätzliche Belastung bekommt.

Zuordnung von Aufträgen zu Systemen eines Sysplex

Ein Youtube Video über die neue Administrator Interface zu dem z/OS finden Sie unter

<http://www.youtube.com/watch?v=8AaDxordGJM>