UNIVERSITÄT LEIPZIG

Mainframe Internet Integration

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

SS2013

Work Load Management Teil 3

Goal Management

Work Load Manager Operation

Die System Management Facility (SMF) Komponente des Workload-Managers sammelt Daten über den aktuellen Zustand aller durch WLM verwalteten Resourcen. Dies sind Information über die Prozessoren, den Hauptspeicher, die Benutzung der I/O Einrichtungen 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 Work Unit festgestellt, ob sie Resourcen benutzt (Using) oder darauf wartet (Delay). Der Workload-Manager verwendet diese Daten, um die Verteilung von Resourcen 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**

Jingkai Chen: Modellierung eines Goal orientierten Workload-Managers. Diplomarbeit Universität Tübingen, November 2006
http://www-ti.informatik.uni-tuebingen.de/~spruth/DiplArb/Chen07.pdf

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

Vor allem für Service Klasseb nit Stapelverarbeitung Work Units macht ein Response Time Goal wenig Sinn. 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 Resourcen 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 Resourcen, 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 Resourcen Zugang für Service Klassen mit Zielvorgaben begrenzt, wenn diese ihre Ziele deutlich übererfüllen.

Response Time Goal

Response Time oder Antwortzeit drückt den Wunsch aus, dass die Zeit, die Work Units (z.B. 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 Mainframe Rechner. Netzverzögerungen werden von WLM nicht erfasst und deshalb auch nicht berücksichtigt.

Execution Velocity Goal

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

Velocity = theoretisch beste Zeit / wirklich verbrauchte Zeit

Execution Velocity ist wie folgt definiert:

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

Näherungsweise ist dies:

$$\text{Velocity} = \frac{\text{Zeit für CPU + I/O}}{\text{Zeit für CPU + I/O + paging + MPL + swapin + queues}}$$

(MPL = Multiprogramming Level)

Nehmen wir z.B. an, dass eine Work Unit insgesamt fünf ms lang auf Resourcen wartet und insgesamt fünf ms lang Resourcen benutzt. Man erhält eine *Execution Velocity* von 50:

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

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

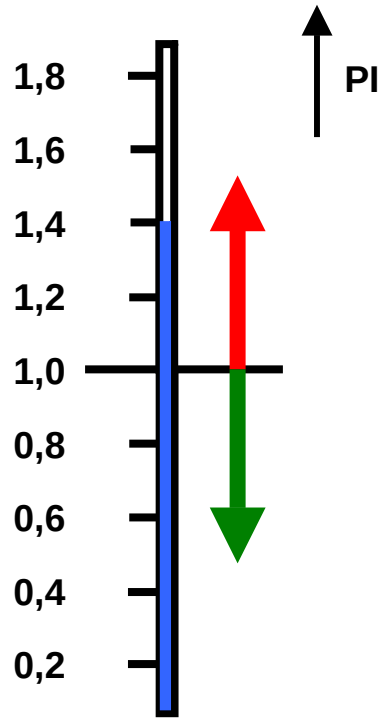
Performance Index

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

Workload-Manager-Algorithmen lösen 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 Resourcen**
PI = 1:	**Ziel wurde exakt erreicht. Alles ist ok**
PI > 1:	**Ziel wurde nicht erreicht. Die Service Class benötigt zusätzliche Resourcen**



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:

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

Definition des Performance Index für das Response Time Goal:

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

Ein PI < 1bedeutet, Ziele wurden erreicht
Ein PI > 1bedeutet, Ziele wurden nicht erreicht

Ein PI = 1 signalisiert Zielerfüllung.

Wichtigkeit (Importance)

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

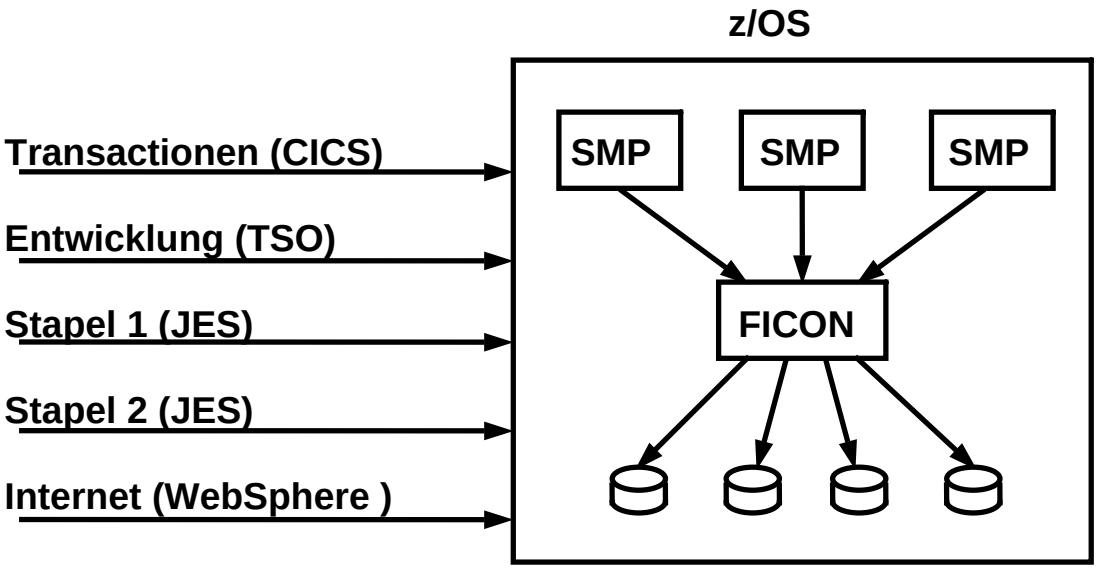
Neben der Zielvorgabe wird jeder Service Class (Dienstklasse) eine Wichtigkeit (Importance) zugeordnet, die festlegt, welche Klassen bevorzugt bzw. benachteiligt werden sollen, wenn benötigte Resourcen in dem Mainframe Server nicht mehr ausreichend zur Verfügung stehen.

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 Resourcenn 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	2

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

A Sample WLM Service Definition

Service Coeeficients	
CPU	1
SRB	1
IOC	0.1
MSO	0.0001

5

Workload	Service Class	Per	Dur	Imp	Type	Pct	Value	# Rules	Goal	Comment
BATCH	BATJESME	1		5	ExVel	20		1	Execution velocity of 20	BATCH JES WORKLOAD MEDIUM
BATCH	BATWLMHI	1		4	ExVel	30		2	Execution velocity of 30	BATCH WLM WORKLOAD HIGH
BATCH	BATWLMLO	1		4	Disc			1	Discretionary	BATCH WLM WORKLOAD LOW
BATCH	BATWLMME	1		5	ExVel	20		1	Execution velocity of 20	BATCH WLM WORKLOAD MEDIUM
DATABASE	DB2DDF	1	500	3	RspTime	80	0.5	1	80% complete within 00:00:00.500	DB2 DDF REQUESTS
DATABASE	DB2DDF	2		4	ExVel	10		1	Execution velocity of 10	DB2 DDF REQUESTS
DATABASE	DB2STC	1		1	ExVel	60		24	Execution velocity of 60	DB2 REGION SERVICE CLASS
ONLINE	CICSIMP	1		2	RspTime	90	0.6	5	90% complete within 00:00:00.600	CICS Important Transactions
ONLINE	CICSLOW	1		3	RspTime	80	1	1	80% complete within 00:00:01.000	Other CICS Transactions
STCTASKS	OMVS	1	5000	2	RspTime	80	1	1	80% complete within 00:00:01.000	UNIX System Services
STCTASKS	OMVS	2		3	ExVel	30		1	Execution velocity of 30	UNIX System Services
STCTASKS	STCDFLT	1		4	ExVel	30		1	Execution velocity of 30	Started tasks other/low
STCTASKS	STCHI	1		2	ExVel	50		11	Execution velocity of 50	Started tasks high
STCTASKS	STCMED	1		3	ExVel	30		25	Execution velocity of 30	Started tasks medium
TSO	TSO	1	8000	3	RspTime	90	0.5	1	90% complete within 00:00:00.500	TSO Users
TSO	TSO	2		4	ExVel	20		1	Execution velocity of 20	TSO Users long running

4

1

2

2

3

1. The most important work are the DB2 address spaces.
   - The work running within the DB2 subsystem will be managed according to the transactions
2. At importance level 2 we have the high important started task, including the CICS regions, and the important CICS transactions (CICSIMP)
3. Some service classes have a second period defined. The second periods have a much less aggressive goal and a lower importance
4. Batch runs at the lowest importance: 4, 5 and DISCRETIONARY
5. The MSO (storage) service coefficient reflects the low relevance of storage consumption. Recommended value is 0.0.

Eine gute Faustformel ist < 30 Service Klassen in einer Installation.

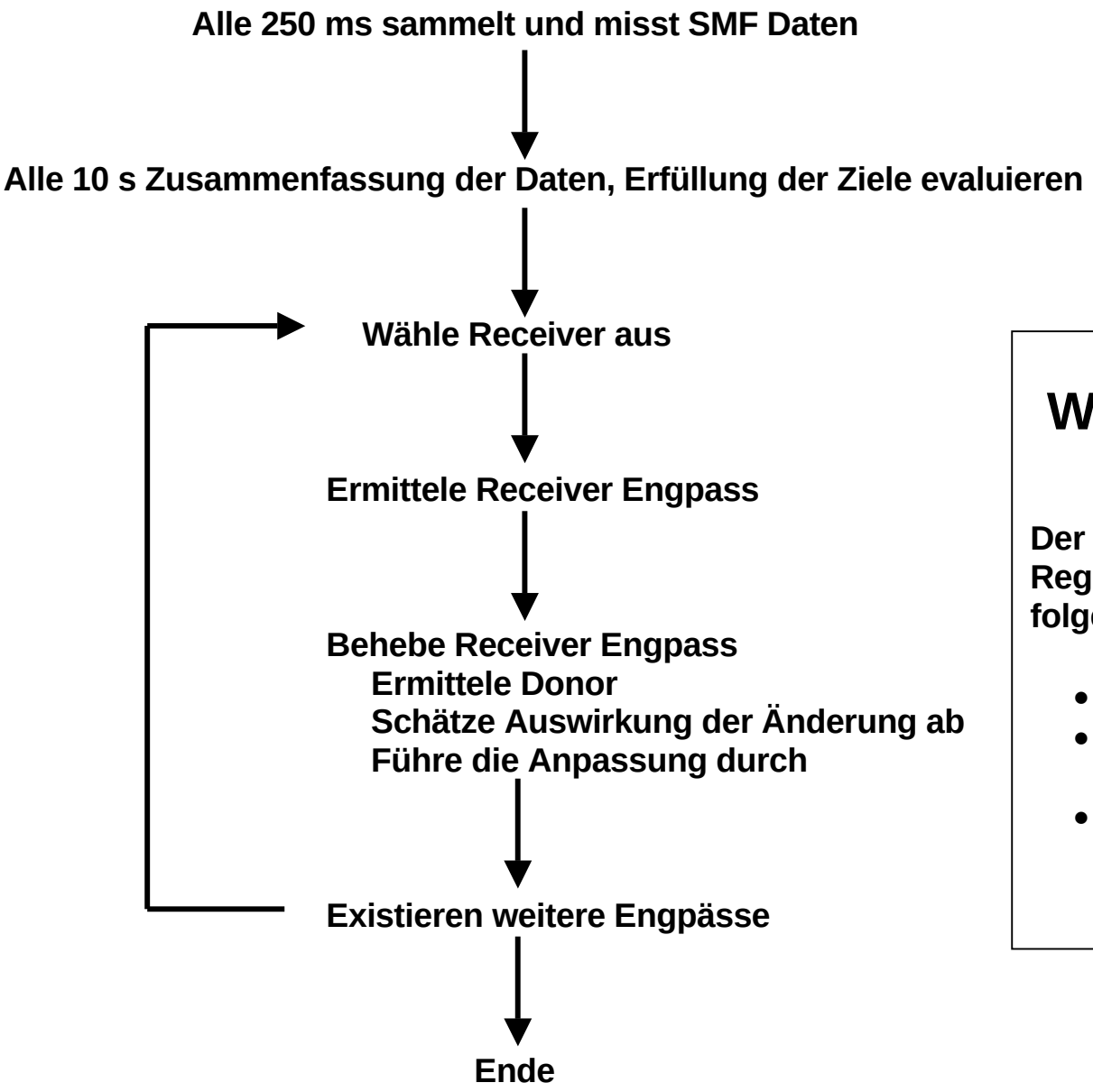
WLM Regelmechanismus

WLM benutzt einen Regelmechanismus, um zur Laufzeit den Zugang zu den Resourcen 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 Work Units auf die Resourcen, die Anzahl der laufenden Work Units und deren Abarbeitungszeiten. Die Informationen werden für die einzelnen Service Classes (Dienstklassen) ermittelt und 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 Resourcen 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 Resourcen abzugeben. Die wichtigere Klasse wird der Empfänger (Receiver) zusätzlicher System Resourcen. Diese müssen natürlich einer anderen Dienstklasse weggenommen werden. Dabei wird allerdings berücksichtigt, ob ein potenzieller Spender (Donor) auch tatsächlich das benötigte Resourcen 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.



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.**

Simulationsmodell

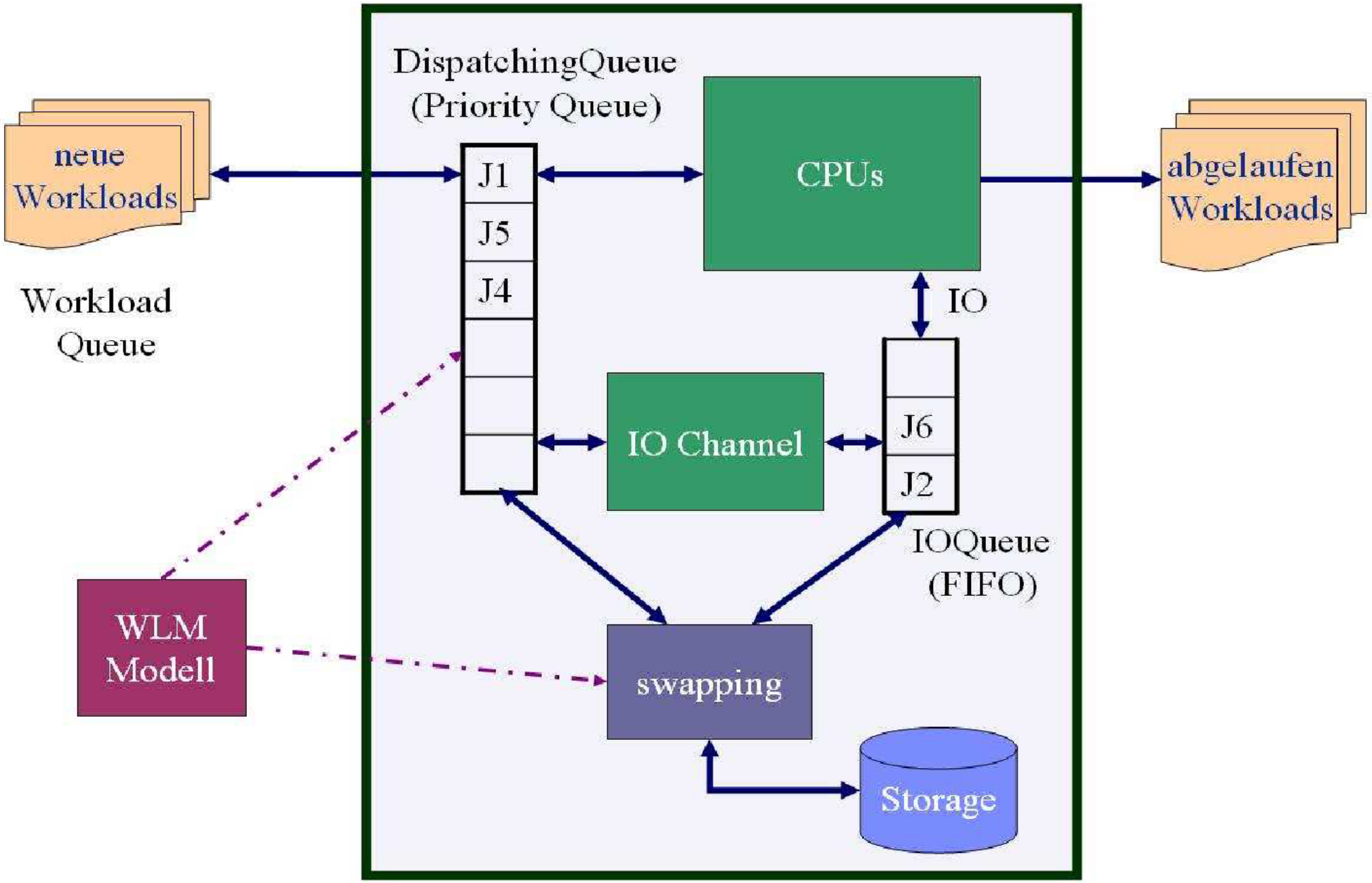
Wir beobachten diese Vorgänge an Hand eines einfachen Simulationsmodells, welcher in der folgenden Abbildung dargestellt ist.

Das simulierte System besteht aus mehren 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 mittels 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 gleichzeitige im System sind, wird 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 auslagert werden und die verbleibenden Adressräume effizienter arbeiten können.


DispatchingQueue
(Priority Queue)
neue
Workloads
Workload
Queue
J1
J5
J4
CPUs
abgelaufen
Workloads
IO
IO Channel
J6
J2
IOQueue
(FIFO)
WLM
Modell
swapping
Storage

Service-Klassen Definition
für das Experiment

Für das Simulationsmodell werden insgesamt 5 Service-Klassen modelliert:

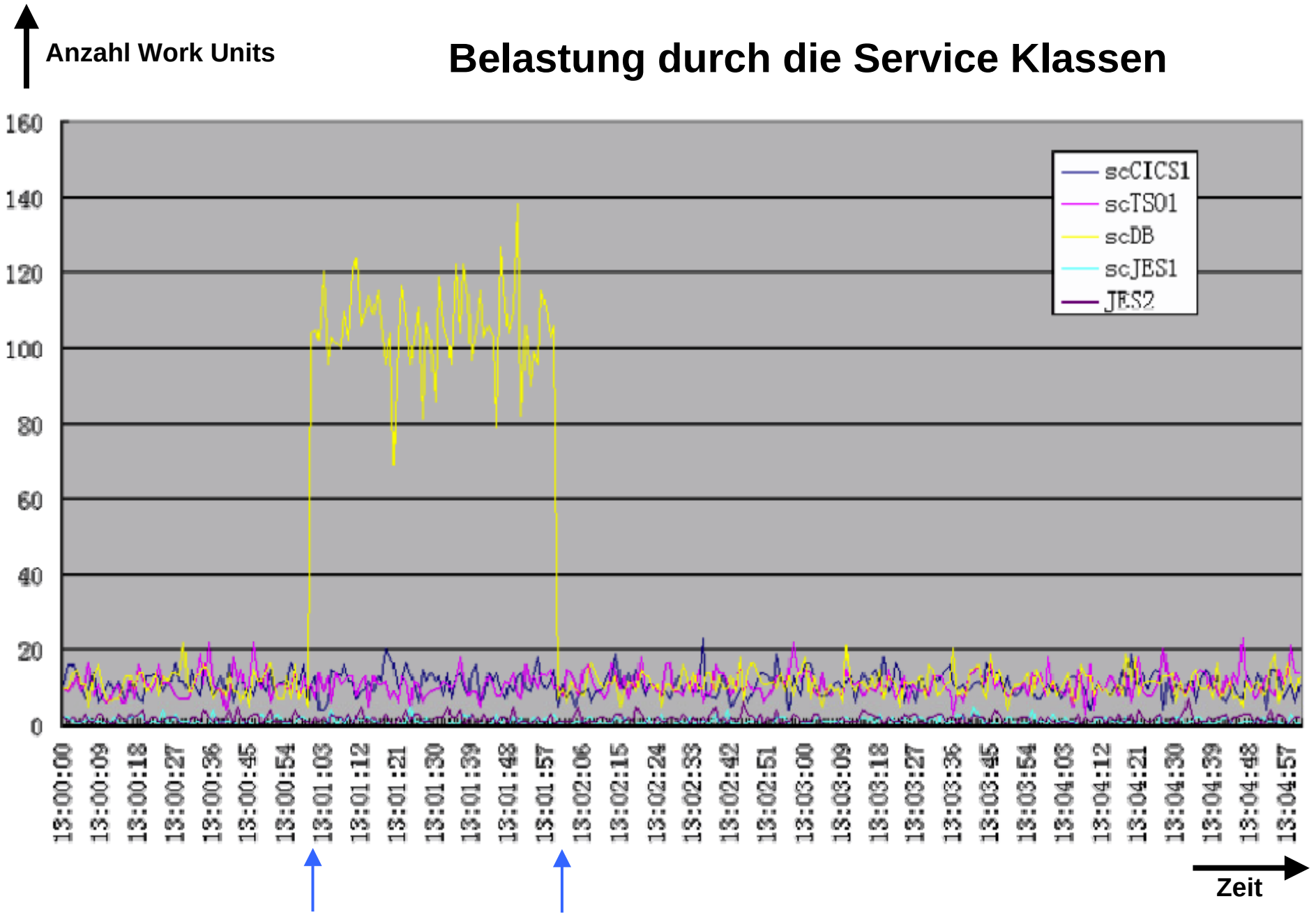
Name	Importance	Goal
scCICS1	1 (hoch)	0,35 s Response
scTSO1	2	0,57 s Response
scDB21	3	0,3 s Response
scJES1	4	15 Velocity
scJES2	5 (niedrig)	15 Velocity

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. Dies ist in der folgenden Abbildung dargestellt.

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

Belastung durch die Service Klassen



WLM arbeitet typischerweise mit 10 Sekunden Intervallen. In diesem Beispiel sind es 9 Sekunden.

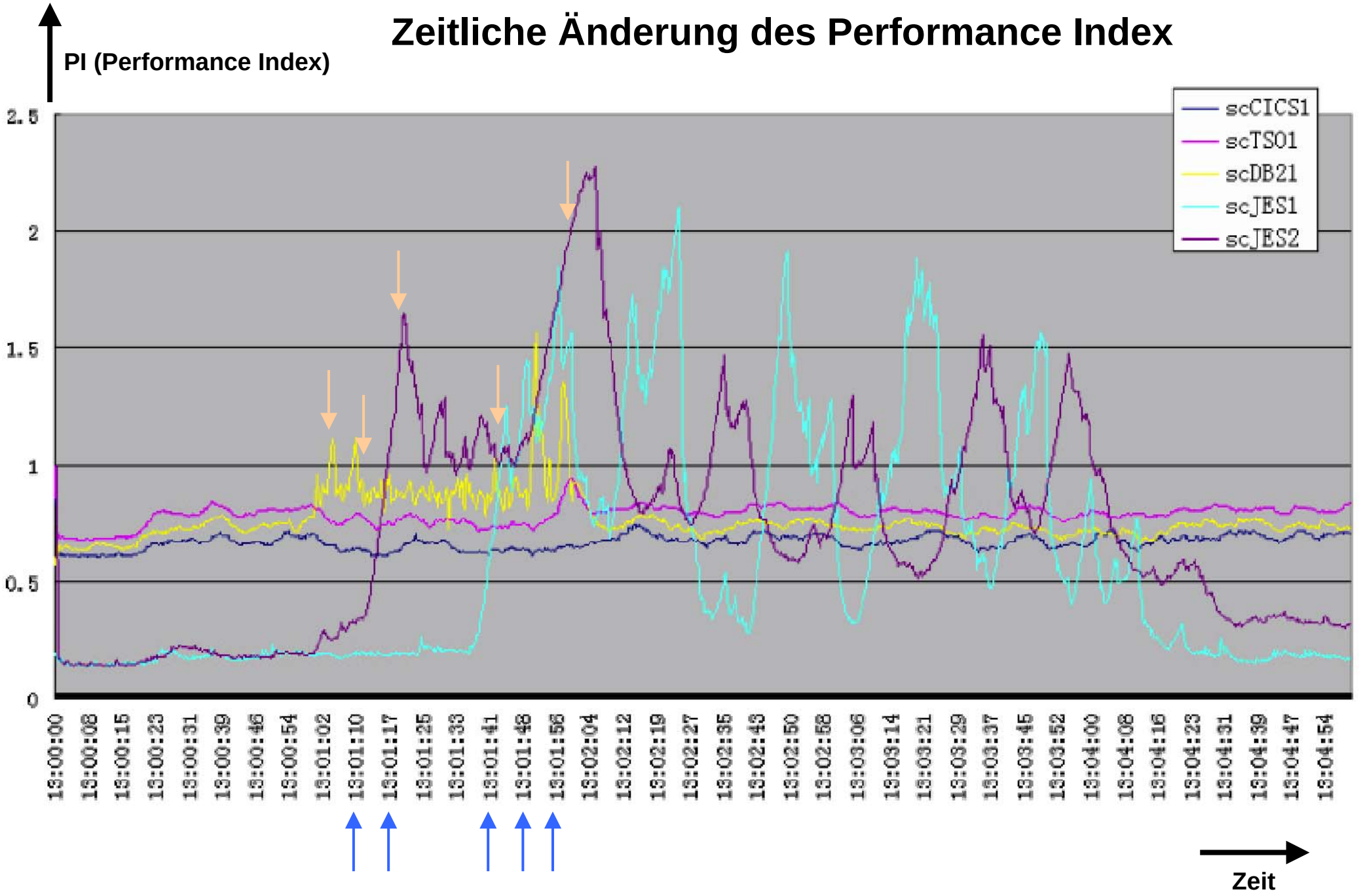
Reaktion des Work Load Managers

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, werde 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 Resourcen,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 scDB2 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 deitlich über 1 liegt. Obwohl die Importance von scJES1 höher als die von scJES2 ist, wird scJES1 zum Donor und gibt Resourcen 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).

Zeitliche Änderung des Performance Index



Zeitlicher Ablauf des Performance Index

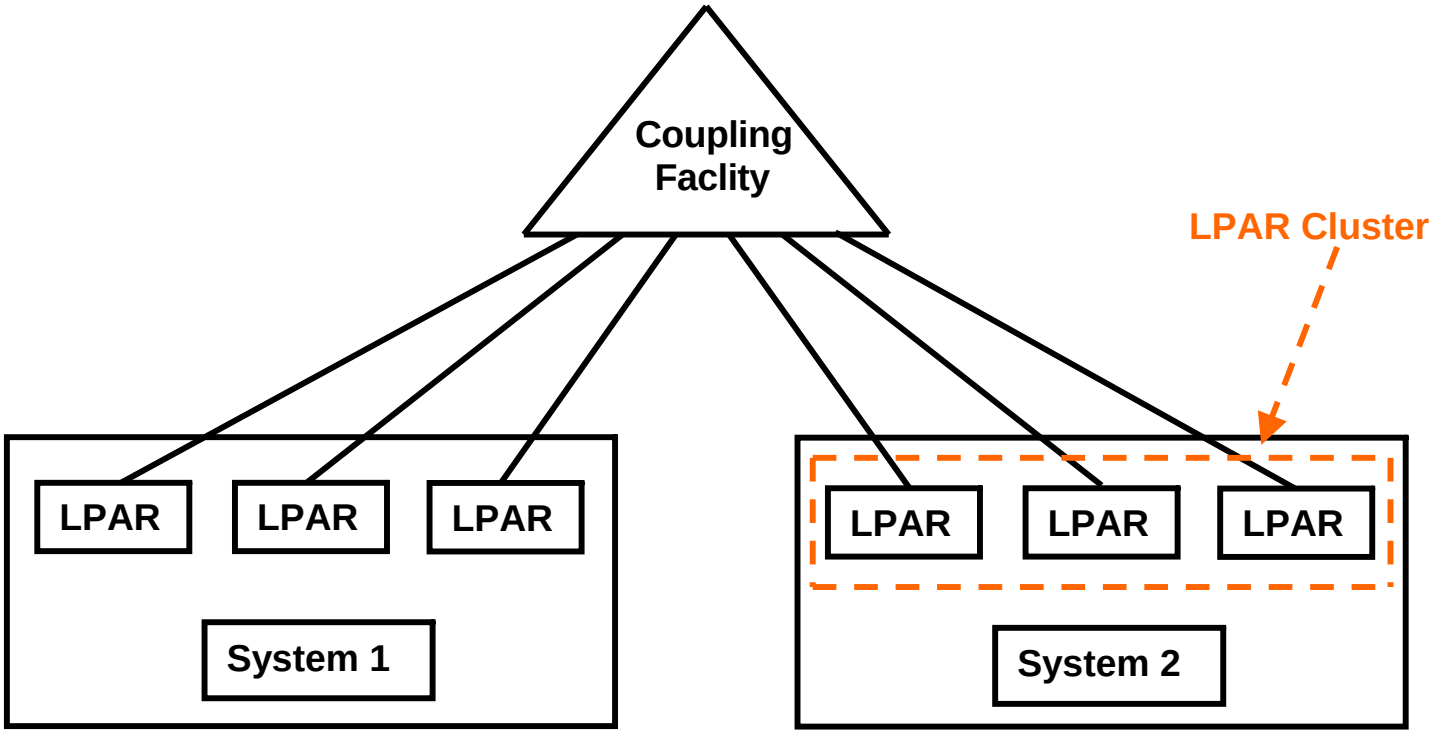
Um 13:01:41 gibt es zwei Service-Klasse scJES1 und scJES2 mit einen PI größer als 1. Der PI von scDB2 liegt unter 1. Service Klasse scDB2 gibt deshalb Resourcen an Service Klasse scJES1 ab. Das Ergebnis ist wenig erfreulich, denn

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 Resourcen 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 Erfahrung liefert WLM deshalb auch in komplexen Installationen sehr befriedigende Ergebnisse.

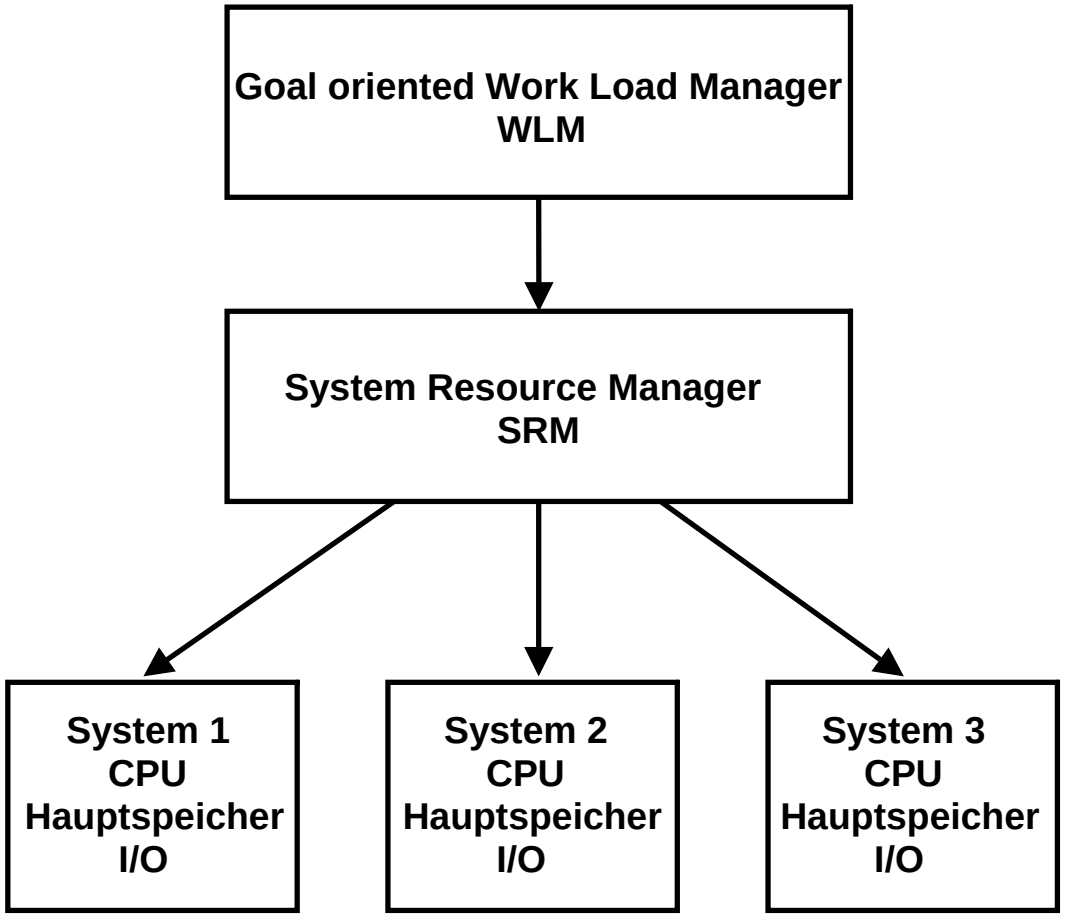


WLM und Sysplex

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.



Der z/OS Work Load Manager kann nicht nur einen einzelnen Rechner, sondern auch einen ganzen Sysplex steuern.

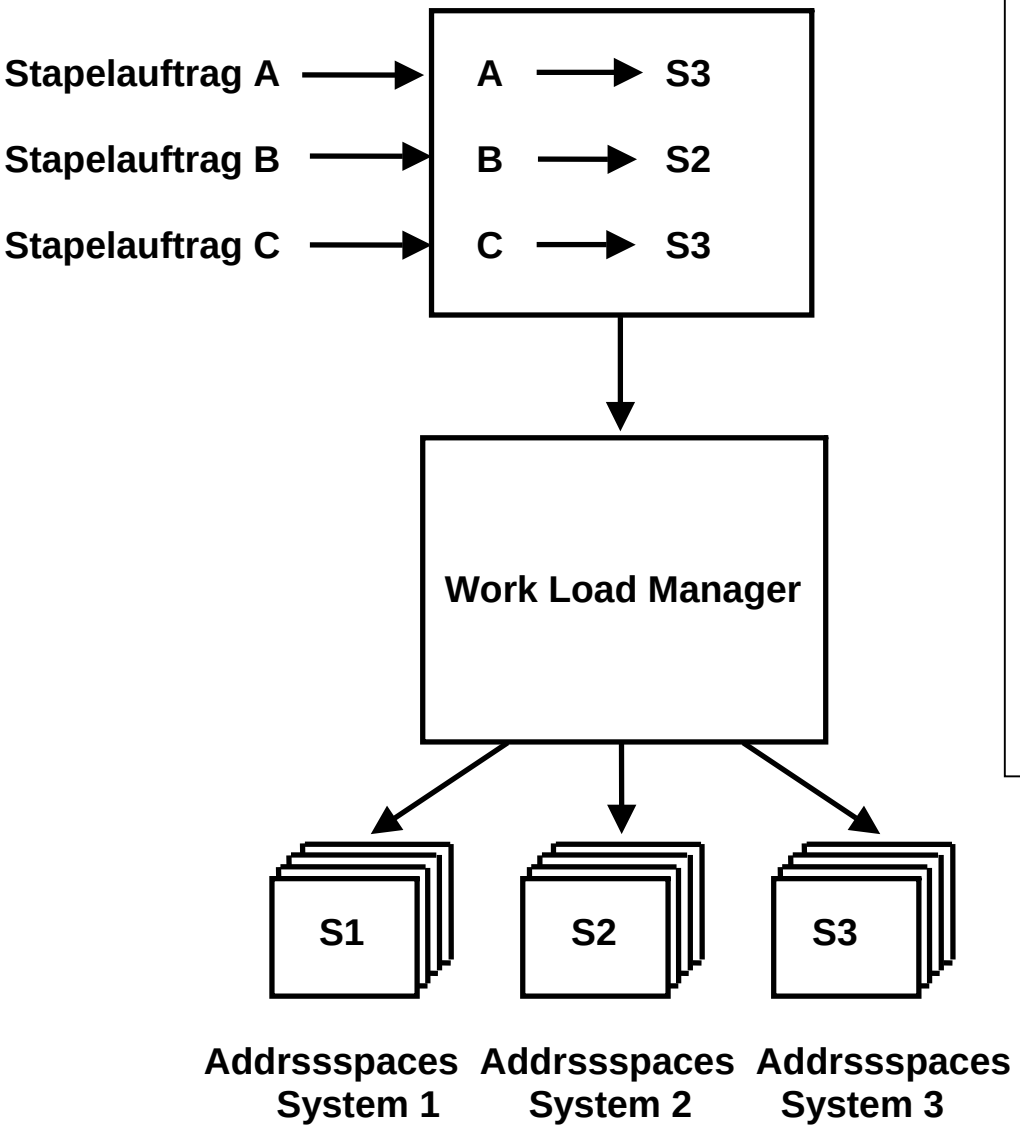
Angenommen ein Sysplex mit drei Systemen.

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

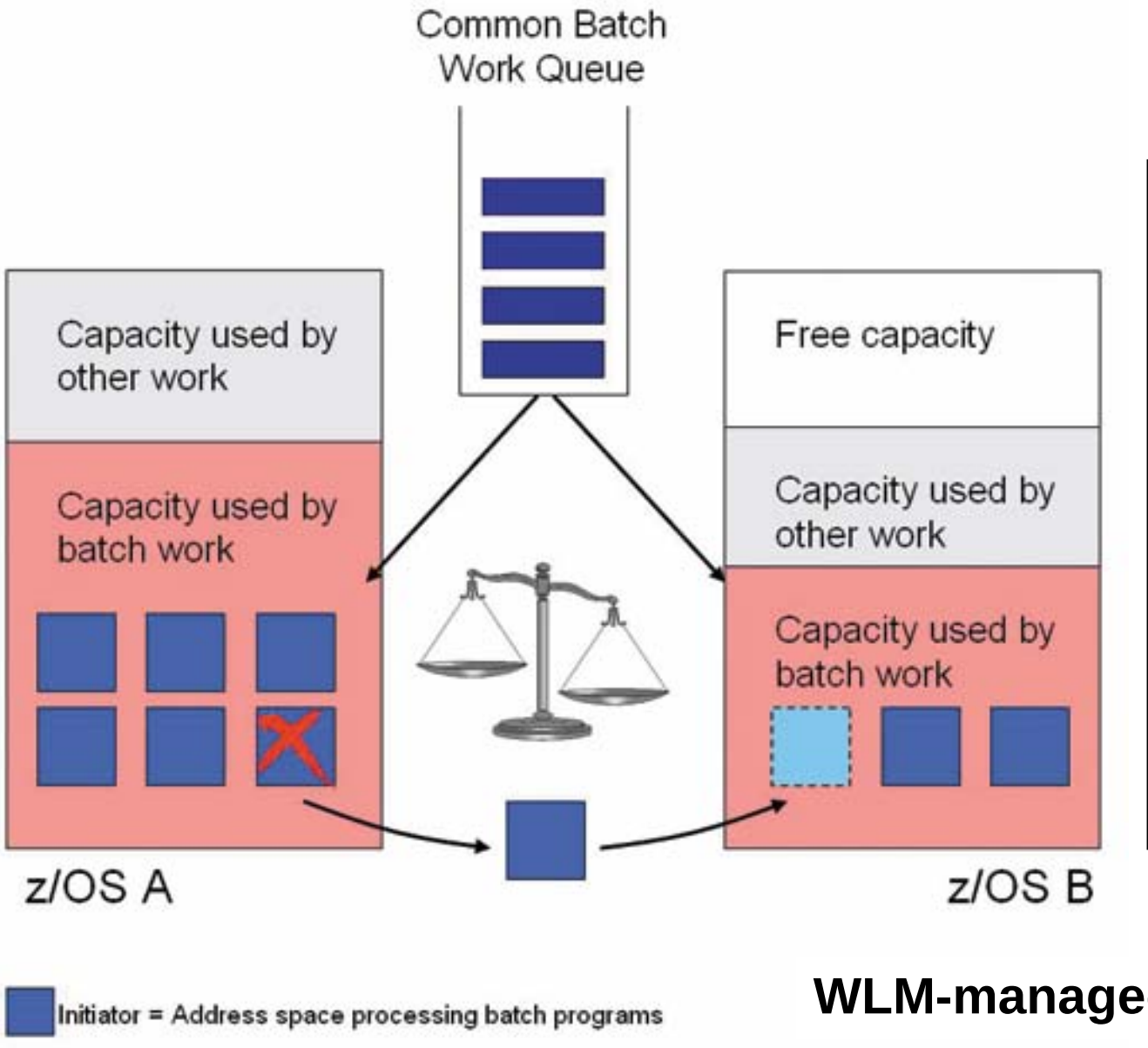


Eine Aufgabe des Work Load Managers ist es, Einstellungen der einzelnen Systeme eines Sysplex dynamisch an sich ändernde Belastungen anpassen und automatisch justieren.

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



Für JES2 entscheidet der WLM (Workload Manager) , wann wie viele Initiators auf welchen Systemen des Parallel Sysplex-Verbundes gestartet und gestoppt werden. In dem Beispiel wird ein Initiator in System A gestoppt und ein neuer Initiator in System B gestartet.

Der WLM stützt sich dabei auf die Vorgaben der Performance Goals in der WLM-Policy.

WLM-managed JES Initiators

Das Moodle Script „Einführung in z/OS“ erläuterte die Funktion der Initiators eines Job Entry Subsystems (JES), siehe http://jedi.informatik.uni-leipzig.de/de/Vorles/Einfuehrung/Bs/zos02.pdf#page=16