13. z/OS Betriebssystem

13.1 Work Load Manager Komponenten

13.1.1 Work Load Management für einen WWW Cluster

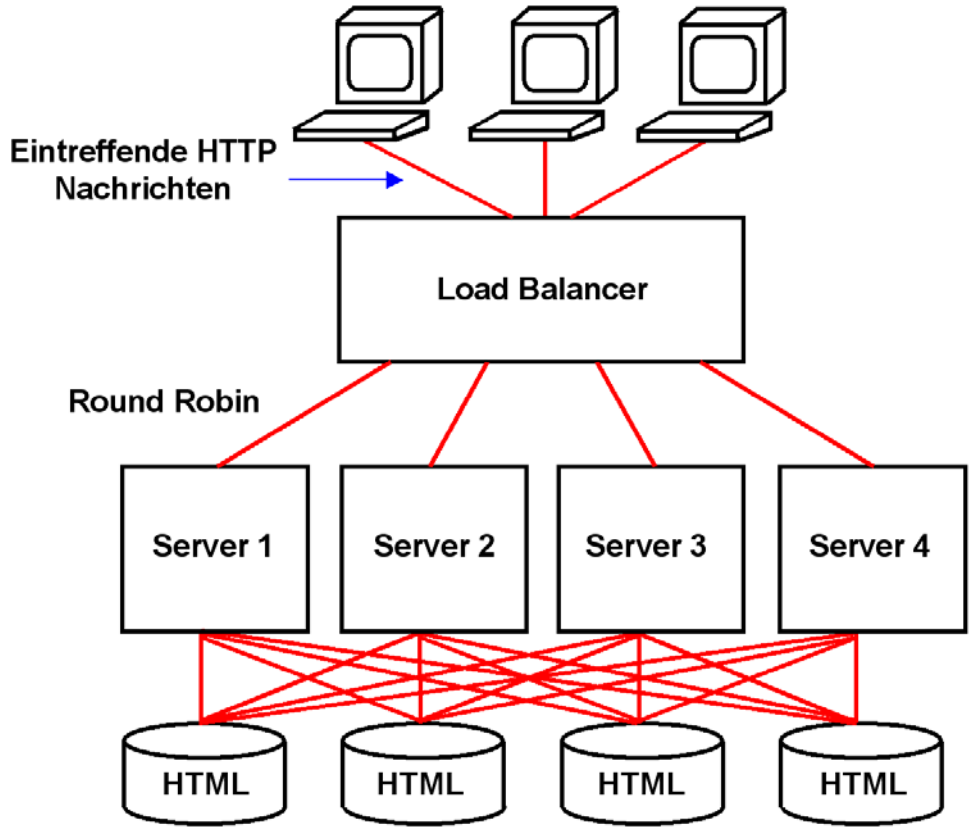

Abb. 13.1.1
Round Robin Routing für statische HTML Server

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.

Beispiel Google: Die Workload Aufteilung ist trivial einfach, weil alle Anfragen identisch sind und nur wenige unterschiedliche Server-seitige Verarbeitungsprogramme benötigt werden. Fast alle Datenbankzugriffe sind Lesezugriffe, und eine Datenintegrität der in mehreren duplizierten Datenbanken abgespeicherten Daten ist nicht erforderlich.

Typische Mainframe Aufgaben wie Datenbanken, Transaktions- und Stapelverarbeitung erfordern sehr viel komplexere Verfahren der Workload Aufteilung.

13.1.2 z/OS Work Load Management Übersicht

Dieser Abschnitt enthält 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 „Work“ (Arbeit) bezeichnet. Der einem bestimmten Prozess zuzuordnende Teil davon wird als Work Unit (Arbeitseinheit) bezeichnet. Die in einem bestimmten Augenblick anfallende „Work Load“ ist die Summe aller Work Units.

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 (Resourcen) notwendig, da viele unterschiedliche Anwendungen den Rechner gleichzeitig nutzen und eine den Benutzerwünschen entsprechende Resourcenverteilung erfolgen muss.

Betriebsmittel sind z.B. CPU Zeit, Hauptspeicherplatz, I/O Kanalkapazität oder Plattenspeicherzugriffe. Auch Netzwerk Adapter oder Locks können Resourcen sein. WLM steuert die Betriebsmittelvergabe auf der Basis von Service Classes (Dienstklassen). Work Units werden den Service Classes 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 (Goal) für die Dienstklassen. Die Zielvorgabe kann die durchschnittliche Antwortzeit der Work Units, die in der Klasse laufen, umfassen, einen prozentualen Anteil der Work Units, die in einer bestimmten Zeit enden sollen, oder eine durchsatzorientierte Vorgabe darstellen. Welches Ziel für eine Service Class vergeben werden kann, hängt davon ab, wie viele Informationen der Workload-Manager über die Anwendungen erhält. Neben der Zielvorgabe wird jeder Dienstklasse eine Importance (Wichtigkeit) zugeordnet, die festlegt, welche Klassen bevorzugt bzw. benachteiligt werden sollen, wenn die Betriebsmittel im System nicht mehr ausreichend zur Verfügung stehen.

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 Work Units auf die Betriebsmittel, die Anzahl der laufenden Work Units und deren Abarbeitungszeiten. Die Informationen werden in Service Classes (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 (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 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 zulassen. Der Zugang zu den Prozessoren wird zum Beispiel über Dispatch Priorities geregelt. Dazu wird allen Arbeiteinheiten einer Dienstklasse (Service Class) dieselbe Priorität zugeordnet, wobei jedoch die Vergabe dieser Priorität nicht in jedem Fall mit der Definition der Wichtigkeit (Importance) 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 (Goal oriented) 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, z.B. Sun M9000 Solaris oder Superdome HP-UX.

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.

Soweit der Wikipedia Beitrag.

Zusammenfassung:

Der Work Load Manager (WLM) ist ein z/OS Alleinstellungsmerkmal. Andere Plattformen, z.B. HP Superdome oder Sun M9000, verfügen ebenfalls eine als Work Load Manager bezeichnete Komponente, die vor allem benutzt wird, um virtuelle Maschinen realen CPUs zuzuordnen. Diese haben jedoch bei weitem nicht den Funktionsumfang des z/OS WLM.

13.1.3 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 einen optimalen Umfang an realem Speicher zuzuordnen. Prozesse können zeitweise ausgelagert werden, wenn Ressourcen knapp werden,**
- **Multithreading, Anzahl von Subsystem Instanzen,**
- **In einem Sysplex können Prozesse bestimmten 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 (von Minute zu Minute, während des Tages, innerhalb einer Woche),**
- **Die Affinität der Prozesse zu ihren Daten soll optimiert werden,**
- **Die Zuordnung der Prozesse zu realen CPUs muss erfolgen,**
- **Prozesse erzeugen unterschiedliche I/O Anforderungen und I/O Belastungen.**

In einer IT-Umgebung ohne Mainframe ist es üblich, unterschiedlichen Anwendungen unabhängige physische Server zuordnen. 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 (siehe Abschnitt 14.3.13) Komponenten beitragen.

Für die weitere Diskussion müssen Sie sich mit einer Reihe von Begriffen vertraut machen:

Work	**andere Bezeichnungen Datenverarbeitung. Data Processing. Ist das, was ein Rechner durchführt.**
Work Unit	**ein Stück Datenverarbeitung, z.B. Durchführung einer Transaktion oder eines Stapelverarbeitungsjobs.**
Work Load	**= ∑ 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	**Resourcenverbrauch 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 eine Service Class, z. B. 90 % aller Transaktionen einer bestimmten Service Class sollen eine Response Time < 0,3 Sekunden aufweisen.**

Unter System Resourcen 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 Resourcen optimal auszunutzen.

Anmerkung: Die deutsche Sprache verwendet die Schreibweise Ressource. Die engliche 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.

13.1.4 z/OS Work Load Manager Komponenten

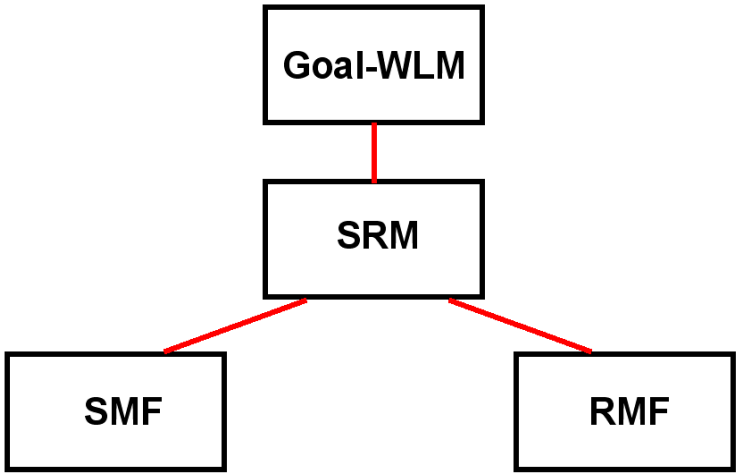

Abb. 13.1.2
Die Komponenten des Work Load Managers

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, welche die eigentlichen Anpassungen vornimmt.**
- **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.**

Im allgemeinen Sprachgebrauch wird unter dem Begriff Work Load Manager (WLM) die Summe aller 4 Komponenten verstanden. Häufig versteht man unter WLM aber auch nur die oberste Komponente, den Goal Oriented Work Load Manager Goal-WLM.

13.1.5 System Resource Manager

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

SRM überwacht die Nutzung der Resourcen 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.

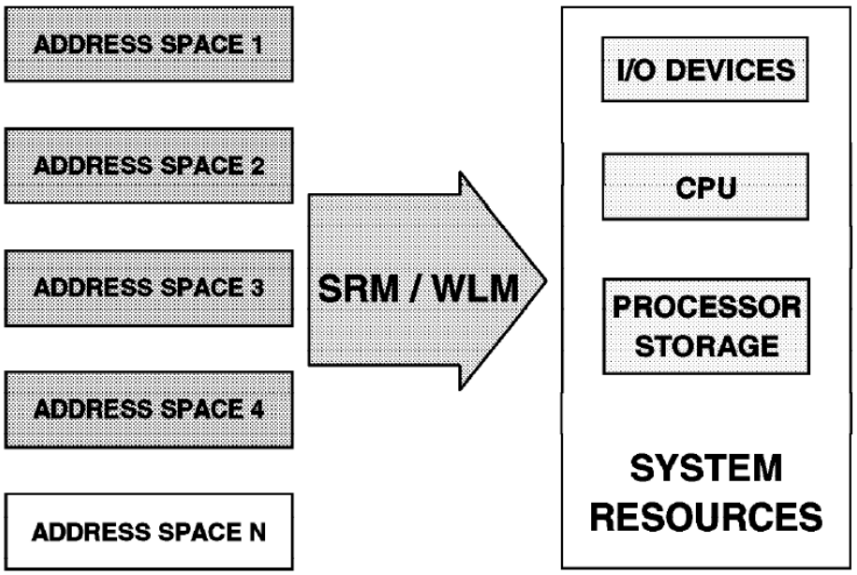

Abb. 13.1.3
System Resources Manager

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 Resourcen (und wie viel 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 Resourcen werden optimal genutzt um einen möglichst hohen Systemdurchsatz zu erreichen.**

13.1.6 System Management Facility

Die System Management Facility (SMF) überwacht kontinuierlich, welche Betriebsmittel eine Service Class (Gruppierung von Work Units 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, Prozessor Aktivitäten und Hauptspeichernutzung für jeden Job Step und jede TSO Session. Hilfsprogramme analysieren die Daten und erstellen Berichte.

13.1.7 Resource Measurement Facility

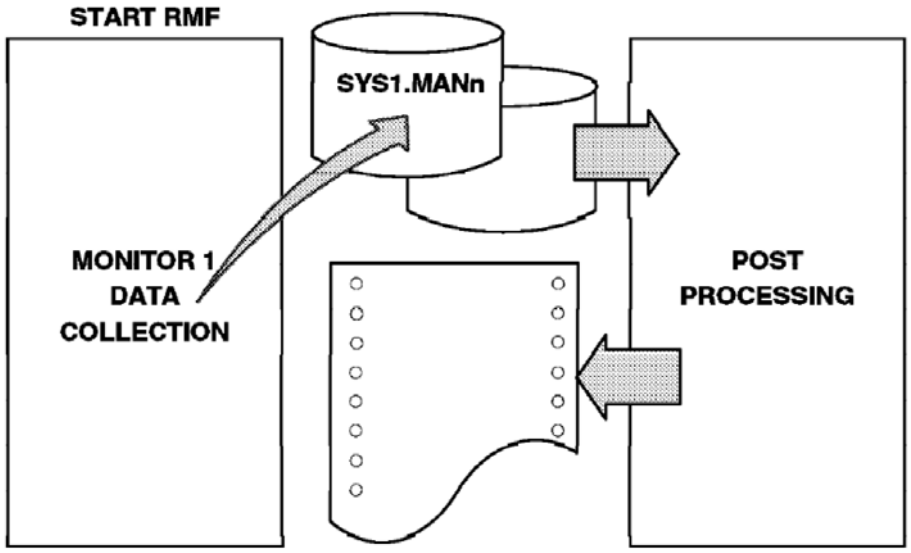

Abb. 13.1.4
RMF Monitor

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.

13.1.8 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 vom Resourcen 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 es, was den z/OS WLM von Work Load Management Funktionen in anderen Betriebssystem unterscheidet.

13.2 System Resource Manager

13.2.1 Resource Management

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

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

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

Der Resourcen Verbrauch eines Prozesses wird in Service Units (Resourcenverbrauch einer Work Unit) gemessen. Es existieren unterschiedliche Service Unit Definitionen für die CPU Zeit, den Hauptspeicherbedarf und die I/O Operationen.

Wir schauen uns das Resourcen Management an Hand der Hauptspeicherverwaltung näher an.

13.2.2 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 Einführung in z/OS, Verarbeitungsgrundlagen, Teil 1, http://jedi.informatik.uni-leipzig.de/de/Vorles/Einfuehrung/Grundlag/vg01.pdf#page=12). Normalerweise steigt die Auslastung der CPUs, je größer die Anzahl der aktiven Prozesse ist.

Auf einem Rechner laufen gleichzeitig zahlreiche Prozesse. Sie wechseln zwischen den Zuständen laufend, wartend und ausführbar. Wir bezeichnen diese Prozesse als aktive Prozesse (siehe Band 1, Abschnitt 2.1.9) Normalerweise steigt die Auslastung der CPUs, je größer die Anzahl der aktiven Prozesse ist.

Jeder aktive Prozess beansprucht realen Hauptspeicher für seine aktiven Seiten (im Hauptspeicher abgebildet). Nur ein Teil der Seiten des virtuellen Speichers ist in jedem Augenblick in Rahmen des realen Hauptspeichers 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-, Linux 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 stehlen sich gegenseitig benötigte Rahmen und können auf Grund zahlreicher Fehlseitenunterbrechungen kam noch produktive Arbeit leisten.

13.2.3 Auslagern von Prozessen

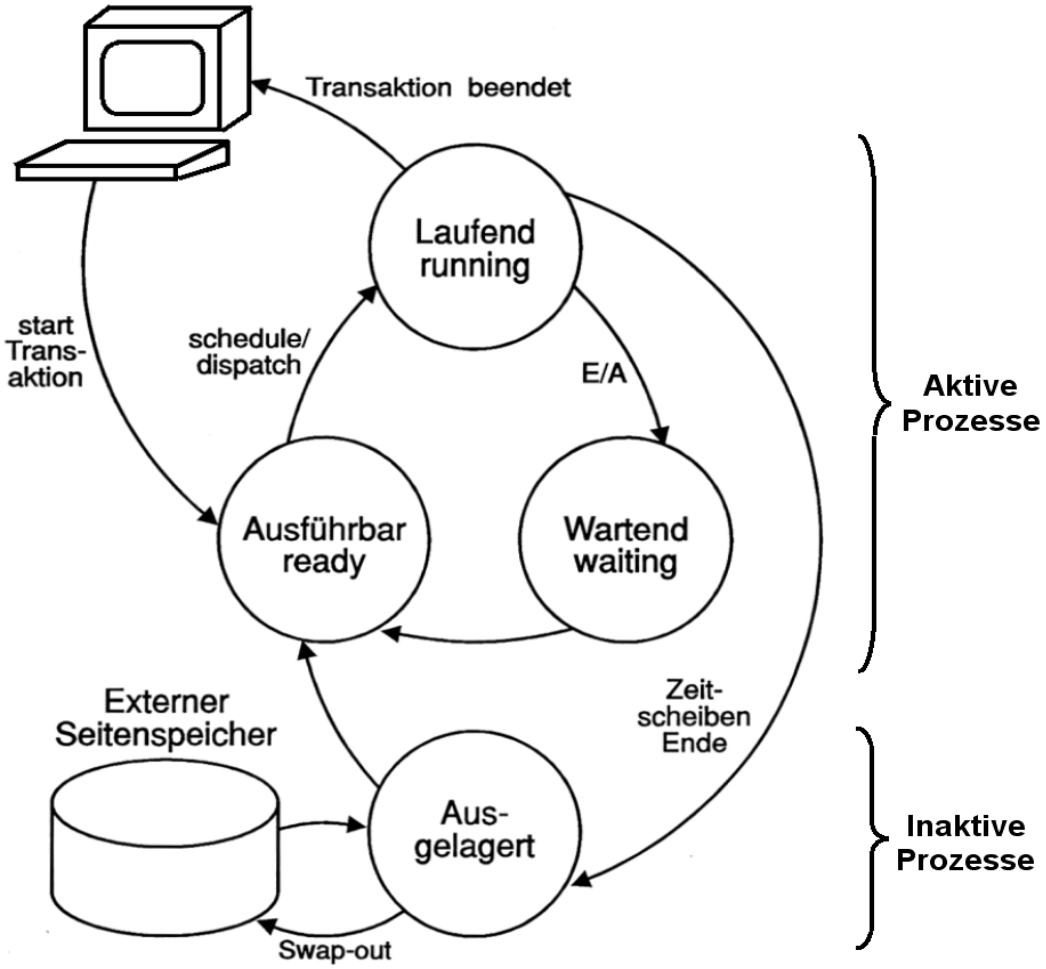

Abb. 13.2.1
Aktive und inaktive Prozesse

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 Resourcen, z.B. CPU Zeit schlecht ausgenutzt.

13.2.4 Begrenzung der Seitenfehlerrate

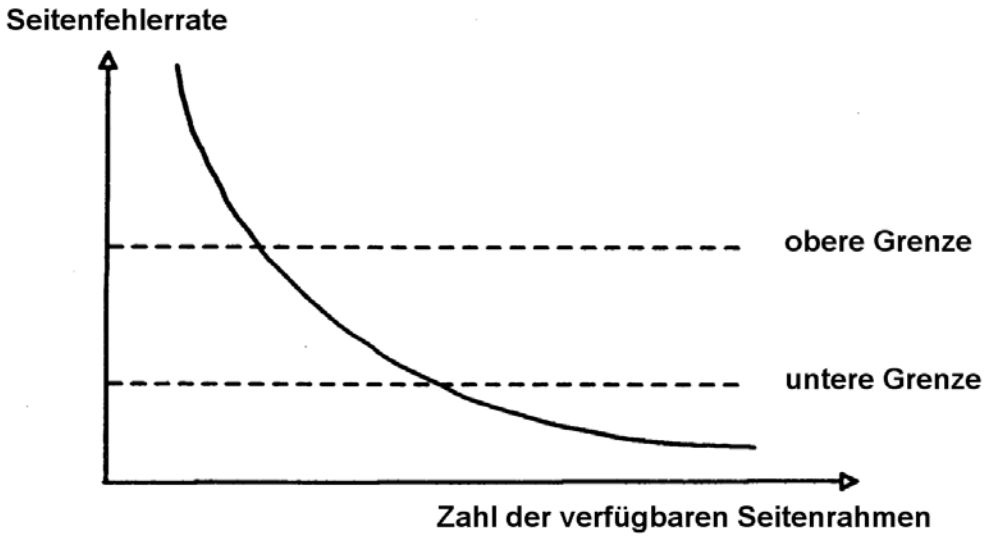

Abb. 13.2.2
Eingrenzung der Seitenfehlerrate

Um Flattern zu verhindern, beobachtet der Workload Manager 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, wird WLM die Anzahl der aktiven Prozesse wieder erhöhen (Swap-in), um die Auslastung der CPUs und/oder anderer Resourcen zu verbessern.

13.2.5 Wirkungsweise des Seitenwechsel-Begrenzers

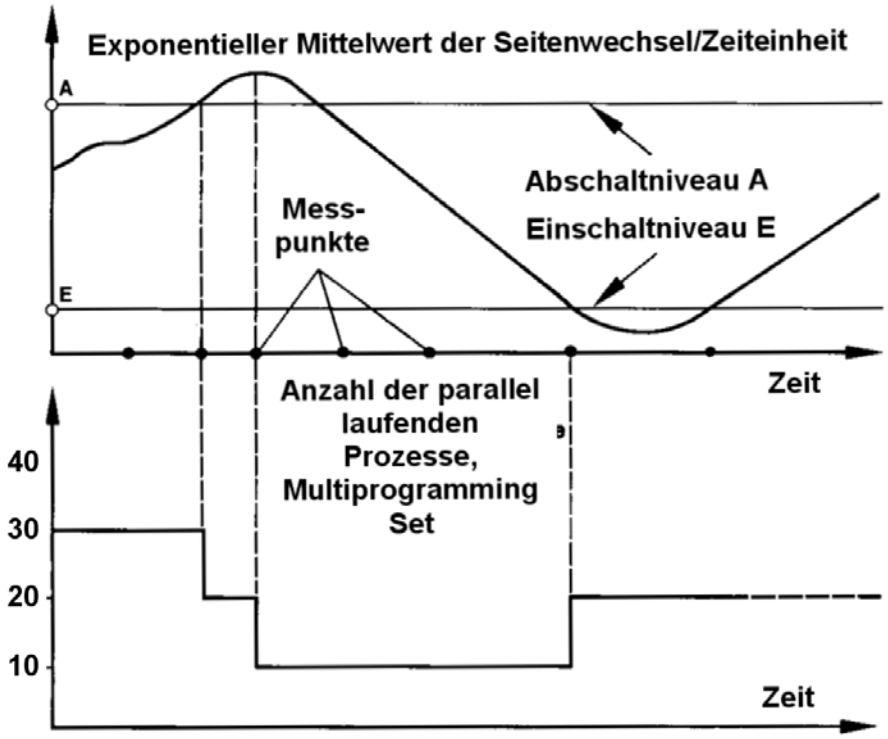

Abb. 13.2.3
Die Multiprogramming Set Größe wird entsprechend der Seitenfehlerrate angepasst

Als Multiprogramming Set wird die Menge der derzeitig aktiven 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.

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

Dieser Wert ändert sich ständig..

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

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

13.2.6 Prioritätssteuerung für inaktive Prozesse

Abb. 13.2.4
Entscheidungsparameter für Swap-in bzw. Swap-out

WLM entscheidet unter Prioritätsgesichtspunkten, welcher aktive Prozess für ein Swap-out selektiert wird bzw. 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.

13.2.7 Automatische Steuerung der System-Resourcen

Die unterschiedlichen Arten von Prozessen haben unterschiedliche Zielsetzungen. So sind z.B. für viele interaktive CICS oder Webserver Zugriffe optimal Antwortzeiten (gemessen in Sekundenbruchteilen) auf Kosten einer optimalen Resourcen 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 einer Stunde, 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 gar nicht mehr schnell genug reagieren.

Immer häufiger tritt auch das Problem auf, dass die Änderung von Systemparametern unerwartete und unerwünschte Seiteneffekte erzeugt. Bei der Vielzahl der Einstellungsmöglichkeiten ist es denkbar, dass eine bestimmte Änderung keine erwartete Verbesserung, sondern eine Verschlechterung bringt.

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

Dies ist die Aufgabe des Goal Oriented Work Load Managers.

Abb. 13.2.5
Zusammenspiel der WLM Komponenten

Vor der Einführung des Goal orientierte Workload-Managers existierte bereits der System Resource Manager (SRM). SRM ist 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 und Auslastung der System Resourcen (z.B. CPU, Hauptspeicher, Ein/Ausgabe).

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

Die Steuerung erfolgte durch die mehr oder weniger statische Eingabe von SRM Konfigurationsparametern durch den Systemadministrator. Diese ließen sich in der Vergangenheit nur schlecht den sich dynamisch ändernden Gegebenheiten anpassen. Weiterhin ist es schwierig, bestimmte angestrebte Ziele (Business Goals, z.B. alle Transaktionen vom Typ x sollen eine Antwortzeit < 0,3 s haben) in SRM Konfigurationsparameter (z.B. Target Multiprogramming Level ≤ y) zu übersetzen.

13.2.8 Goal Orientierung

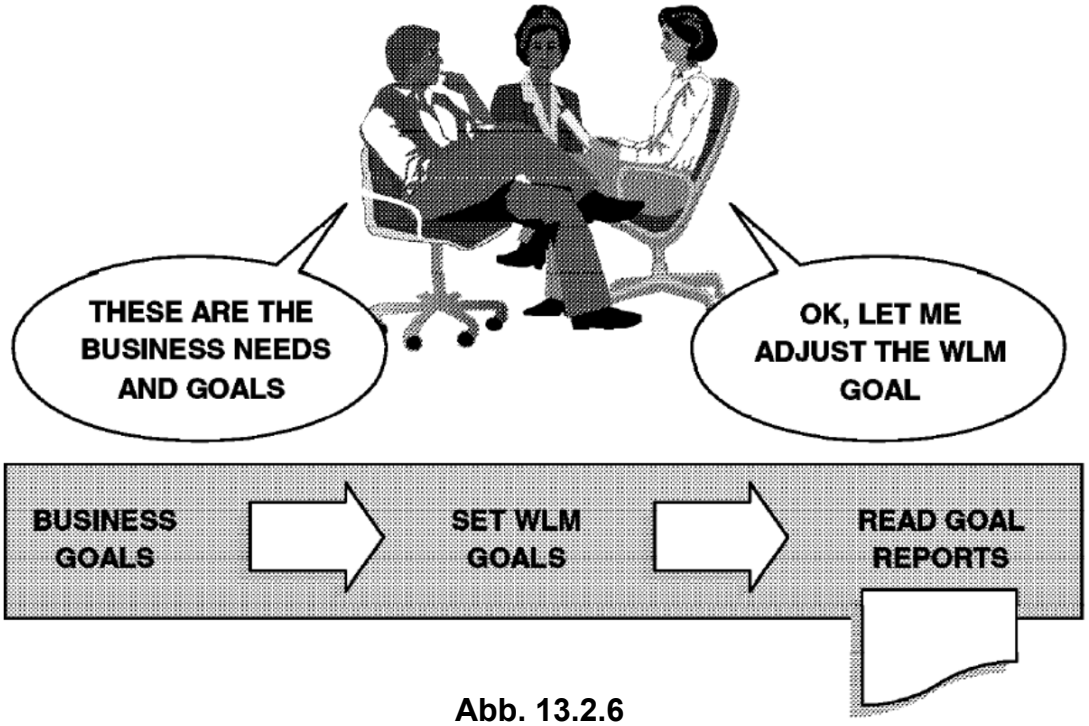

Abb. 13.2.6
Die Festlegung der Goals erfolgt unabhängig von Kenntnissen über die Systemstruktur

Der Goal oriented Work Load Manager (WLM) reduziert den Aufwand für den System Administration und die Notwendigkeit für Detailwissen

Der Goal oriented Workload Manager 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, die von menschlichen Benutzern gewünschte Optimierungs-Anweisungen in der Form von Geschäftsbegriffen (Business Goals, wie z.B. erwünschte Antwortzeit) entgegen zu nehmen. Das Leistungsverhalten der Installation wird durch Vorgaben für Zielsetzungen (Business Goals) festlegt. Der Goal oriented Workload Manager übersetzt diese Anweisungen in System Resource Manager (SRM) Parameter und passt sie dynamisch an die sich laufend ändernde Systembelastung an.

z/OS Subsysteme brauchen zusätzliche Funktionen, um mit WLM kooperieren zu können. Der Goal oriented Workload-Manager unterstützt beispielsweise folgende Subsysteme :

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

Zur Integration der Subsysteme enthalten manche Subsysteme ihre eigenen Work Load Management 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 (distributed WebSphere) unter Anderem dadurch, dass alle z/OS WLM Funktionen in Anspruch genommen werden können.

13.2.9 Service Klassen

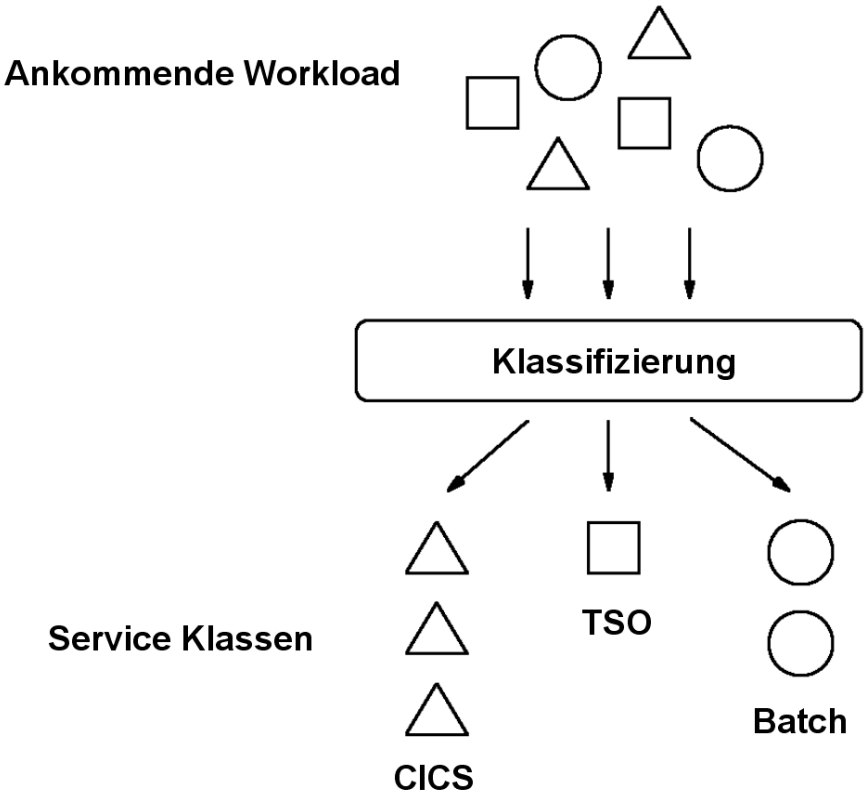

Abb. 13.2.7
Einordnung aller Work Units in Service Klassen

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 Augeblick zahlreiche Work Units parallel verarbeitet. Eine Work Unit wird normalerweise durch einen Prozess ausgeführt. CICS Transaktionen sind ebenfalls Work Units. Bei der Stapelverarbeitung werden Work Units als Jobs bezeichnet.

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 Resourcen 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.

Service-Klassen (Dienst Klassen) sind Einheiten von Workloads mit ähnlichen Charakteristiken, für die die Zielvorgaben definiert werden. Eine mögliche - sehr einfache - Aufteilung der Work Units in Service-Klassen könnte z.B. drei 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 (Dienst Klassen) sind Einheiten von Workload mit ähnlichen Charakteristiken, für die die Zielvorgaben definiert werden.

Die Service-Klasse stellt das Grundkonstrukt für den Goal oriented Workload Manager dar. dar. Sie ist das Ergebnis der Klassifizierung der Workloads mit unterschiedlichen Leistungsmerkmalen in Gruppen, die mit den gleichen Ziel-Vorgaben versehen werden können.

Die Anzahl der Service Klassen sollte nicht zu niedrig (schlechte Optimierung) und nicht zu hoch (hoher Administrationsaufwand und WLM Verarbeitungsaufwand) sein. Eine Anzahl von 25 - 30 Service Klassen ist in vielen Installationen ein guter Wert.

13.2.10 Einordnung in Service-Klassen

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

- **User ID**
- **Art des aufgerufenen Prozesses (Transaktionstyp, Stapel,)**
- **Standort oder Art des Terminals oder Klientenrechners (z.B. Standort Personalabteilung, Art Autorisierung = xxx)**
- **Accounting Information**
- **.....**

Jeder Serviceklasse sind „Ziele“ zugeordnet

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

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

- **Während einer Periode sind begrenzte Resourcen 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 platziert Arbeitsanforderungen so, dass die Wahrscheinlichkeit, alle Ziele zu erreichen, optimiert wird.

Die Einordnung aller Service Units in Service Klassen wird dem System in der Form einer Workload-Manager „Service Definition“ mitgeteilt. Hierzu benutzt der System Administrator eine speziellen administrative Anwendung, die „WLM Administrative Application“. 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 (important) sie ist, um die übergeordneten Geschäftsziele zu erreichen. Entscheidend ist, dass manche Service-Klassen weniger wichtig eingestuft werden als andere. 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 Resourcen geregelt/optimiert werden muss.

13.3 Goal Management

13.3.1 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.cedix.de/DiplArb/Chen07.pdf

13.3.2 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 Klassen mit 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.

13.3.3 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 können von WLM nicht erfasst, und deshalb auch nicht berücksichtigt werden (obwohl dies oft erwünscht wäre).

13.3.4 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 5 ms lang auf Resourcen wartet und insgesamt 5 ms lang Resourcen benutzt. Man erhält eine *Execution Velocity* von 50:

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

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

13.3.5 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**

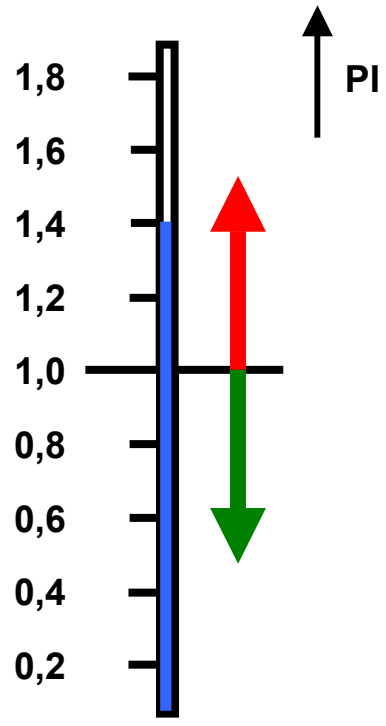

Abb. 13.3.1
Performance Index

13.3.6 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.

13.3.7 Importance (Wichtigkeit)

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 Resourcen 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 Serviceklassen aufgeteilt. Hierbei werden die Stapelverarbeitung-Work-Units in zwei Gruppen mit unterschiedlichen Zielsetzungen aufgeteilt.

CICS Work Units
TSO Work Units
Stapel 1
Stapel 2
Internet

Abb. 13.3.2 zeigt ein Beispiel für die Classification Rules.

13.3.8 Classification Rules Beispiel

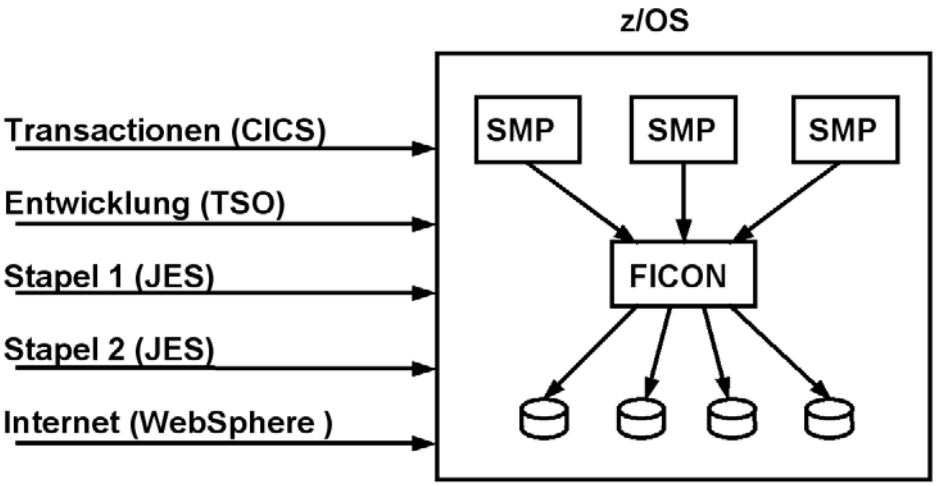

Abb. 13.3.2
Aufteilung der Workload in 5 Service Klassen

	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.

Workload	Service Class	Per	Dur	Imp	Type	Goal Pct	Goal 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	3000	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

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

Abb. 13.3.3
Beispiel einer mit der „WLM Administrative Application“ erstellten Service Definition

13.3.10 WLM Regelmechanismus

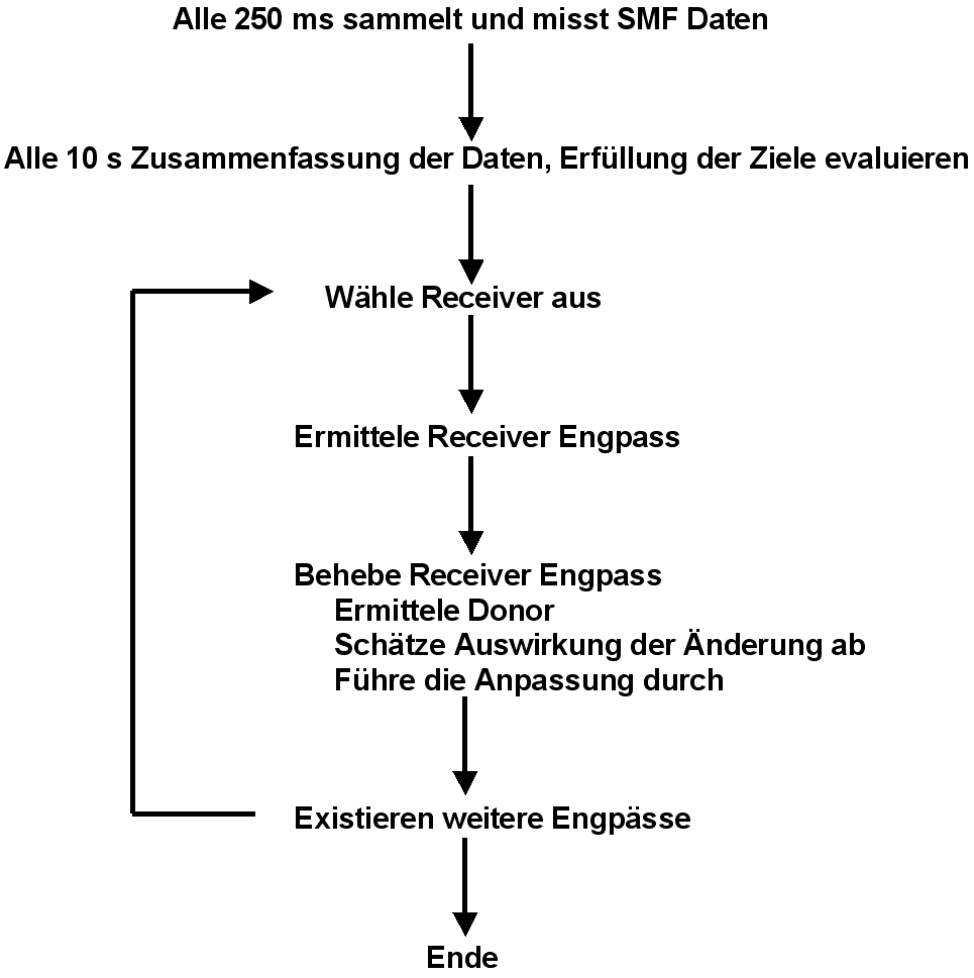

Abb. 13.3.4
Ermittelung von Donor und Receiver

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.

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.**

13.3.11 Simulationsmodell

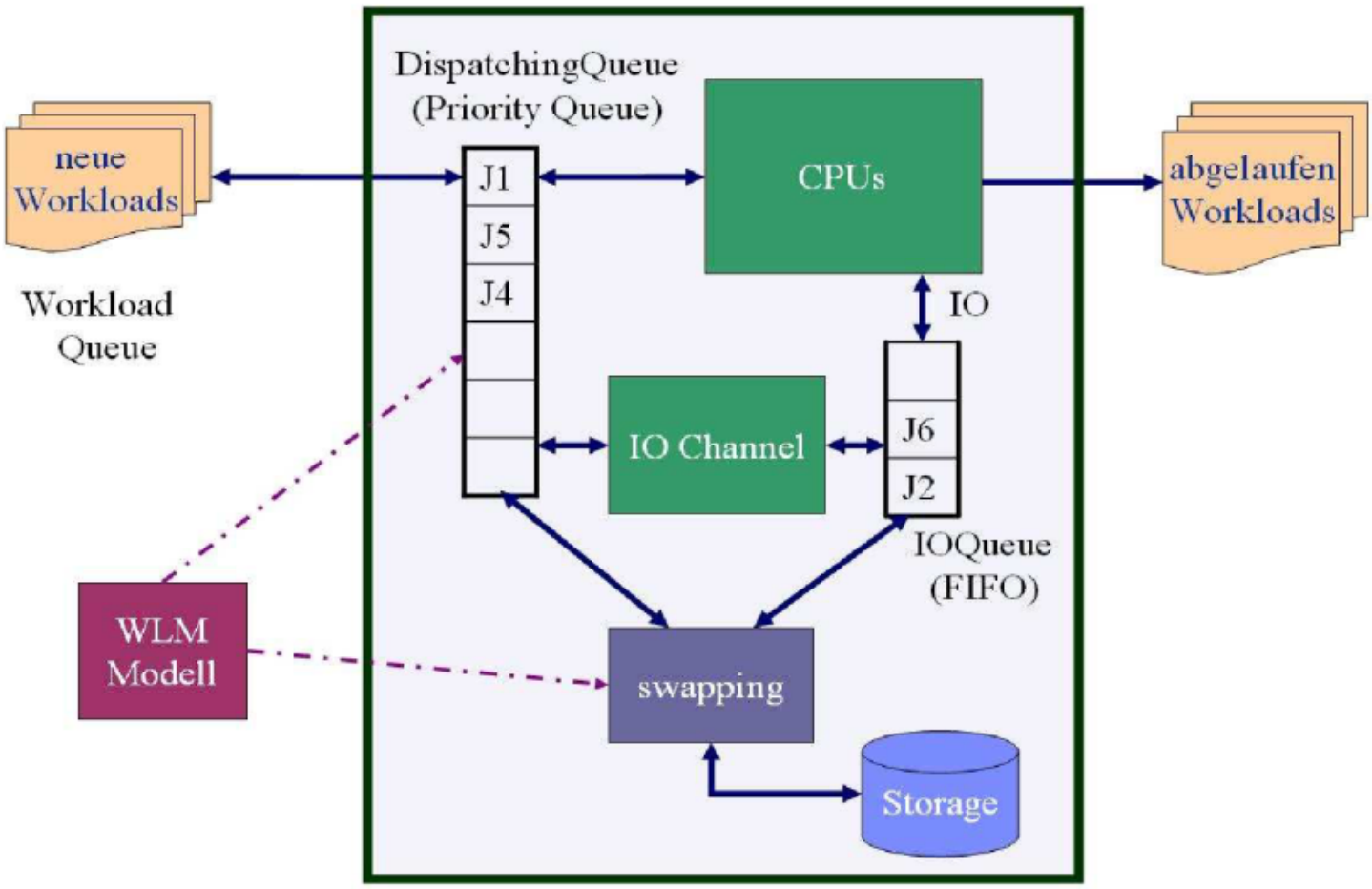

Abb. 13.3.5
Vereinfachte Konfiguration für das Simulationsmodell

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 I/O Queue. 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 Prozess Scheduler ordnet immer dem ersten Prozess in der Dispatch-Queue mit der höchsten Priorität eine 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 die MPL der Service-Klasse reduzieren, damit Adressräume auslagert werden und die verbleibenden Adressräume effizienter (weniger Fehlseiten Unterbrechungen) arbeiten können.

13.3.12 Service 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 Abb. 3.3.6 dargestellt.

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

13.3.13 Belastung durch die Service Klassen

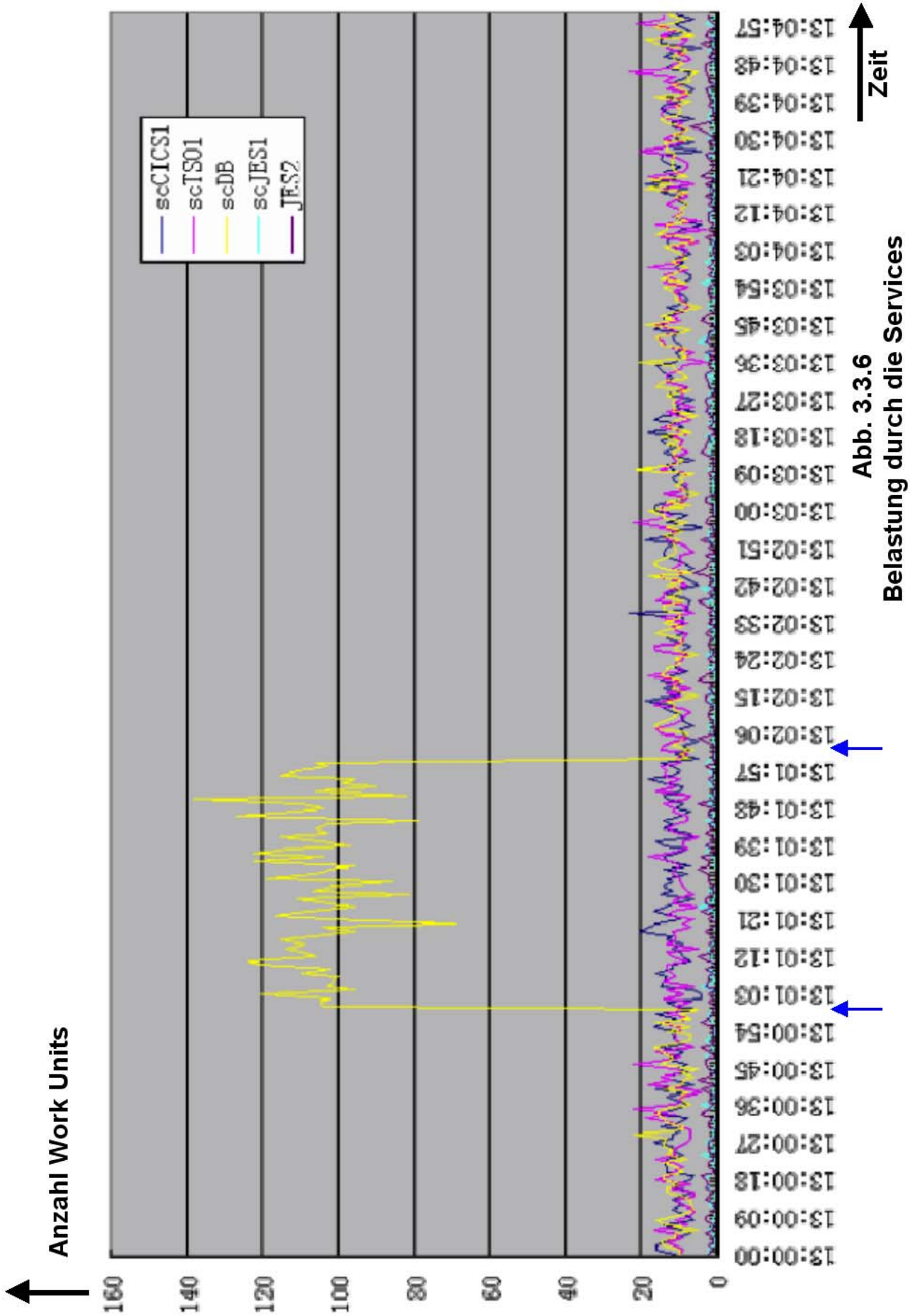

Abb. 3.3.6
Belastung durch die Services

13.3.14 Reaktion des Work Load Managers

In Abb. 3.3.7 ist der zeitliche Verlauf der PIs (Performance Indizes) für die 5 Service Klassen scCICS1, scTSO1, scDB21, scJES1 und scJES2 dargestellt. WLM arbeitet typischerweise mit 10 Sekunden Intervallen. In diesem Beispiel sind es 9 Sekunden.

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 deutlich 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).

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.

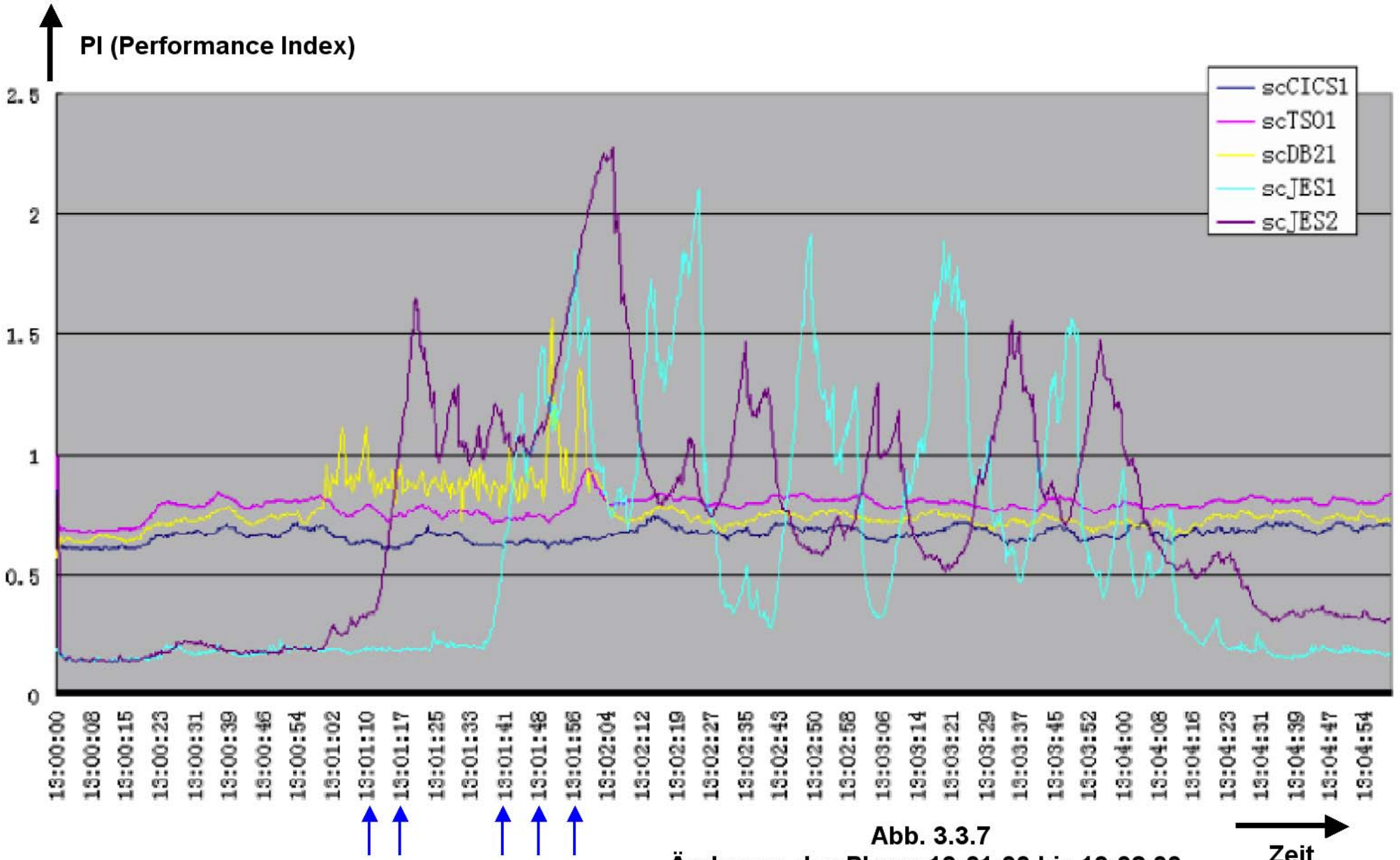

Abb. 3.3.7
Änderung des PI von 13:01:00 bis 13:02:00

13.3.15 WLM und Sysplex

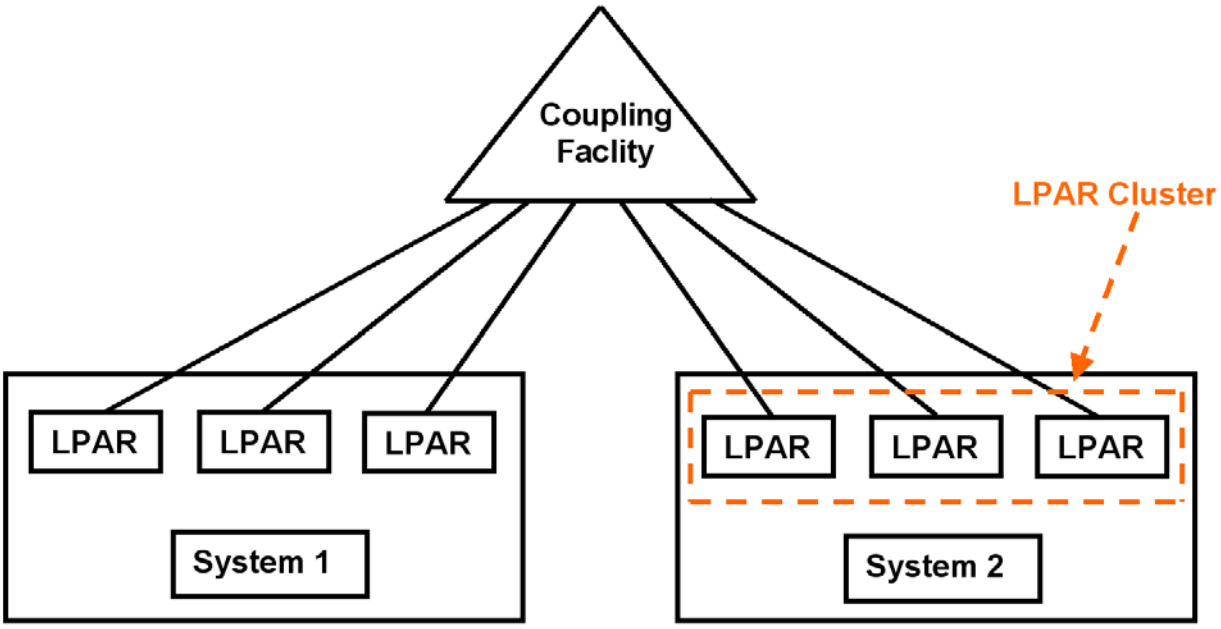

Abb. 13.3.8
Die LPARs eines LPAR Clusters können von WLM gemeinsam gesteuert werden

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.

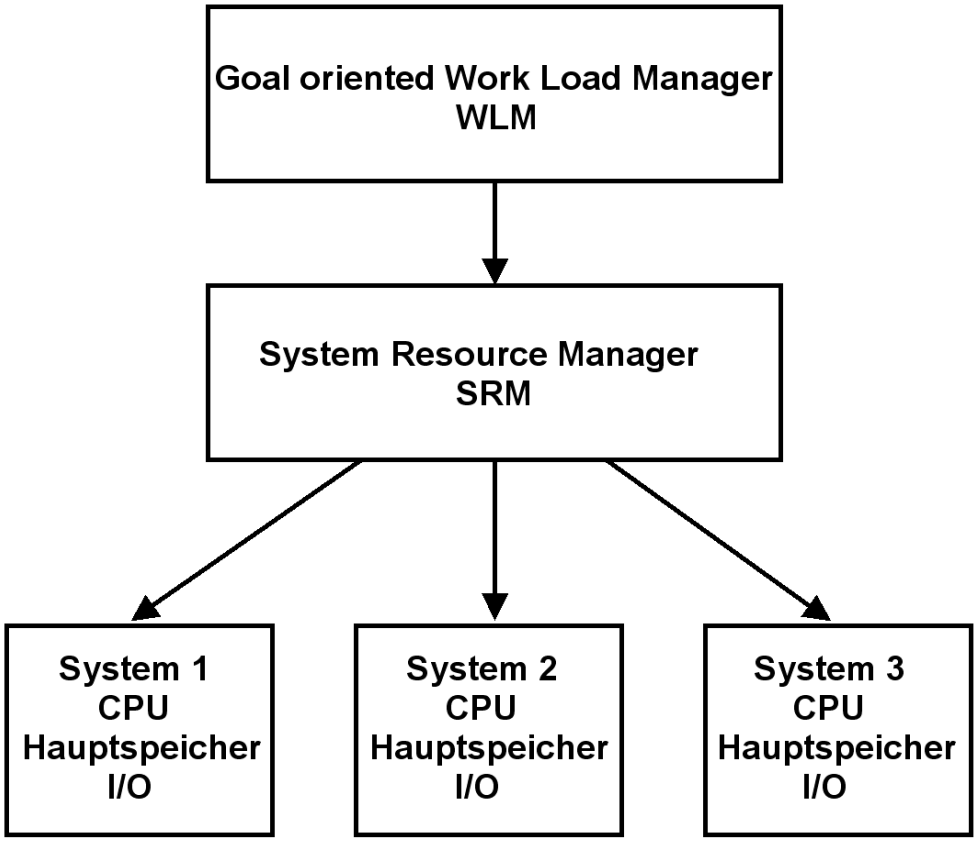

Abb. 13.3.9
Die Systeme (Knoten) eines Sysplex Clusters werden von WLM gemeinsam gesteuert

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.

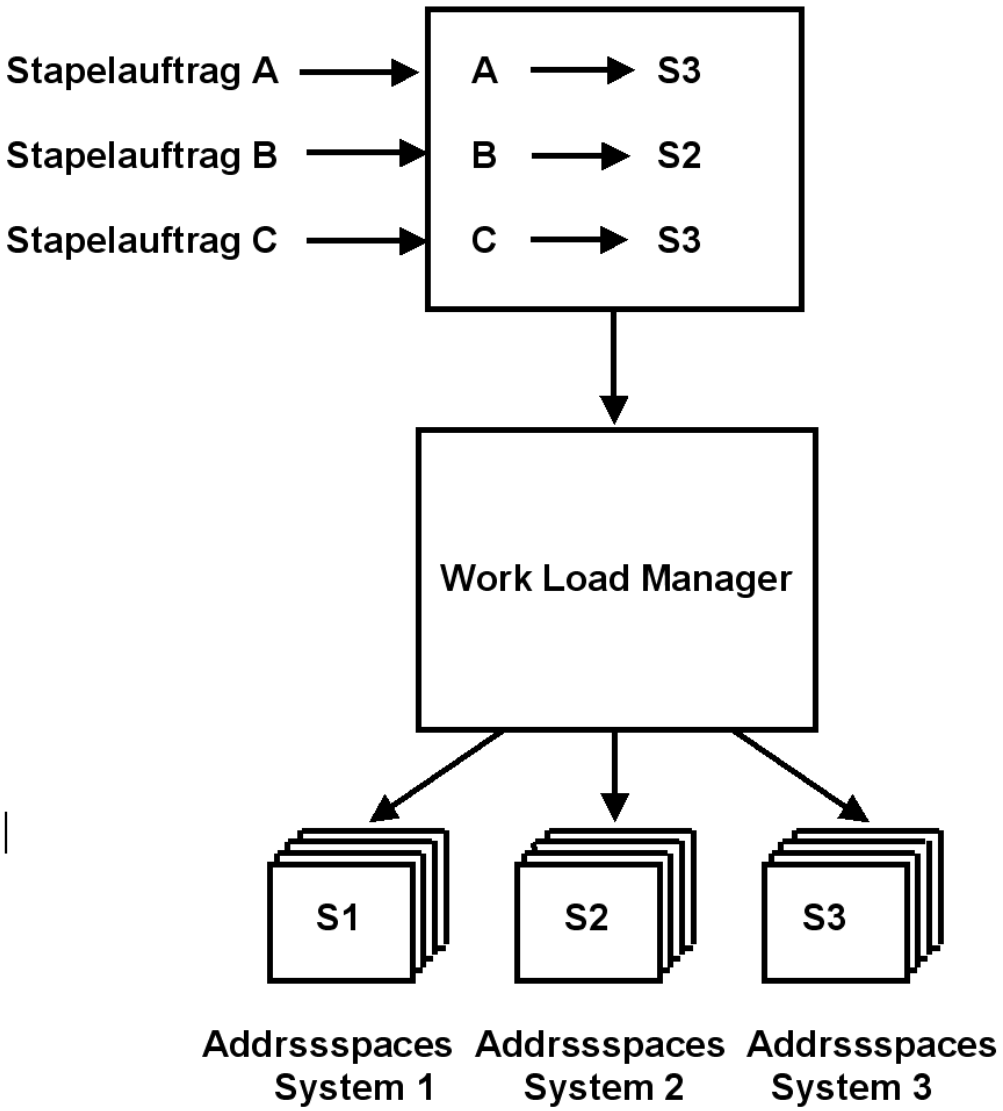

Abb. 13.3.10
Zuordnung von Aufträgen zu Systemen eines Sysplex

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.

13.3.16 WLM-managed JES Initiators

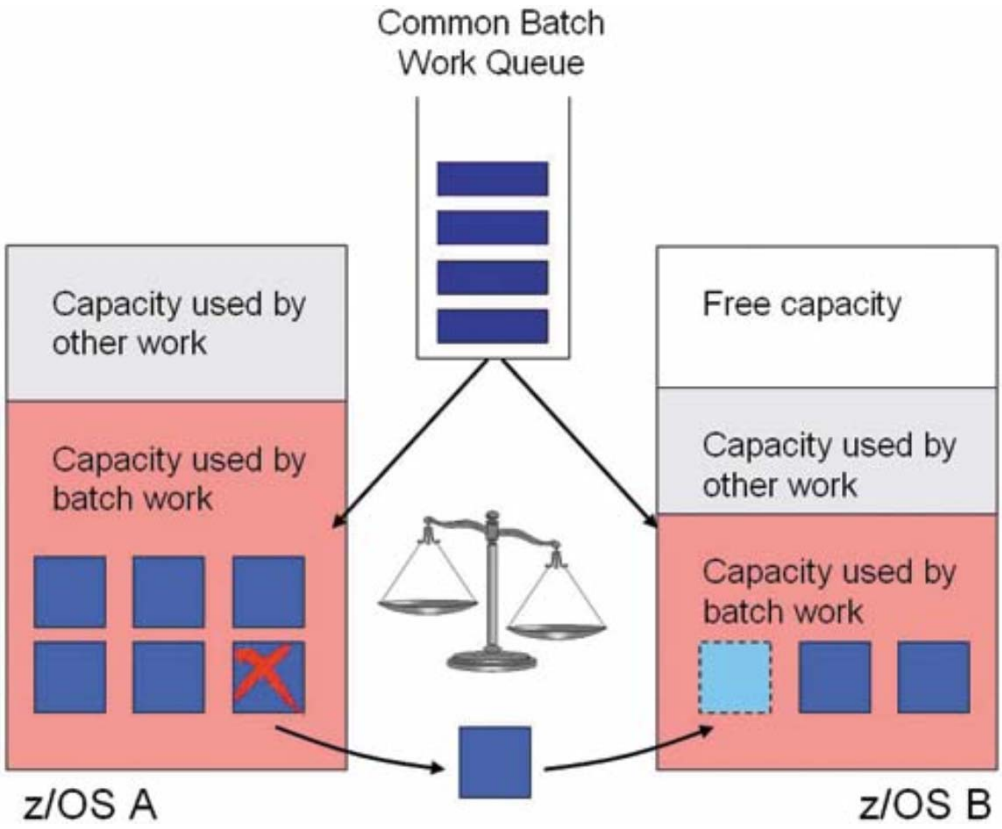

Abb. 13.3.11

Band 1, Abschnitt 3.2.4 erläutert die Funktion der Initiators eines Job Entry Subsystems (JES).

Für JES 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.

13.4 Weiterführende Information

13.4.1 Die wichtigste 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 wichtigsten Redbooks sind

z/OS Intelligent Resource Director, August 2001, SG24-5952-00,
http://www.redbooks.ibm.com/abstracts/sg245952.html

System Programmer’s Guide to: Workload Manager, January 2006, SG24-6472-02
http://www-ti.informatik.uni-tuebingen.de/~spruth/edumirror/xx044.pdf

http://www.redbooks.ibm.com/redbooks/pdfs/sg246472.pdf

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

13.4.2 Videos

Ein Youtube Video über die neue Administrator Interface zu dem z/OS finden Sie unter
http://www.youtube.com/watch?v=8AaDxordGJM

http://www.mdug.org/Presentations/Ed%20Woods%20-%20DB2%20&%20WLM.pdf
enthält einen leicht verständlichen Übersichtsvortrag zum Thema WLM.

http://www.slideshare.net/Tess98/managing-your-db2-for-zos-workloads-using-wlm

13.4.3 Weitere Literatur

WLM ist ein komplexes Thema. Weitere Weiterführende Literatur finden Sie unter:

- **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/sg247083.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/bjnkmstr128.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/sg247083.html**
- **Workload Manager http://de.wikipedia.org/wiki/Workload_Manager**
- **ABCs of z/OS System Programming Volume 12 http://www.redbooks.ibm.com/abstracts/sg247621.html**
- **M. Bensch, D. Brugger, P. Baeuerle, W. Rosenstiel, M. Bogdan, W. Spruth: "Self-Learning Prediciton System for Optimisation of Workload Managememt 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**

13.4.4 Tivoli

Unter dem Oberbegriff Tivoli bietet IBM eine umfangreich Software Suite zur Verwaltung von Informationssystemen an. Sie dienen dazu, Rechner zu überwachen, Software zu verteilen, Systeme zu inventarisieren oder Daten zu sichern. Die Tivoli Software deckt die folgenden Bereiche ab:

- **Operational Management**
- **Service Management**
- **Process Management**

13.4.4.1 Operational Management

Die im Bereich Operational Management zusammengefassten Produkte stellen die Verfügbarkeit der Systeme sicher. Es wird eine Verwaltung, Überwachung und Optimierung der gesamten IT-Landschaft ermöglicht. Genauer werden die Produkte in vier Kategorien unterschieden:

Business Automation

- **System Automation**
- **Provisioning Manager**
- **Intelligent Orchestrator**
- **Monitoring**
- **Workload Scheduling**
- **Business Systems Manager**
- **Service Level Advisor**

Storage Management

- **Tivoli Storage Manager**
- **Total Storage Productivity Center**
- **SAN Volume Controller**

Security

- **Identity Manager**
- **Access Manager for eBusiness**
- **Access Manager for Enterprise Single Sign-On**
- **Security Operations Manager**
- **Security Compliance Manager**
- **Compliance Insight Manager**

13.4.4.2 Service Management

Die IBM Service Management Platform ermöglicht es, automatisch alle über ein Netzwerk erreichbaren IT-Assets (i.a. Geräte der Informationstechnik) eines Unternehmens aufzufinden, zu inventarisieren und zu katalogisieren. Über einen festzulegenden IP-Adress-Bereich wird die Konfiguration eines jeden Konfiguration Items aufgedeckt und gespeichert. In der CCMDB werden diese Informationen abgelegt. Das Change Management ist mit eingeschlossen. Die Produkte sind

- **Tivoli Application Dependency Discovery Manager (ITADDM) und**
- **Tivoli Change and Configuration Management Database (CCMDB).**

13.4.4.3 IT Process Management

Das IT Process Management umfasst vordefinierte Prozesse die auf der Plattform des Websphere Process Managers zur Anwendung gebracht werden können. Basis dieser Prozesse ist die IT Infrastructure Library (ITIL). Die darin etablierten Workflows werden praktisch umgesetzt, um Unternehmen bei der Einführung von ITIL Practices zu unterstützen. Angebotene Process Manager sind

- **Storage Process Manager,**
- **Release Process Manager und**
- **der Unified Process Composer, um weitere Prozesse entwerfen zu können.**

http://de.wikipedia.org/wiki/Tivoli_%28IBM%29

13.4.4.4 Tivoli für z/OS

Die Tivoli Software Suite deckt das gesamte IT Spektrum ab. Nur ein Teil der Produkte läuft unter z/OS oder hat einen unmittelbaren Bezug zu z/OS. Dies gilt besonders für:

- **Tivoli System Automation für z/OS ist eine Policy-basierte, selbstheilende und hochverfügbare Lösung, um die Effizienz und Verfügbarkeit von kritischen Systemen und Anwendungen zu maximieren.**

- **Tivoli NetView für z/OS bietet Automatisierung, Netzwerk- und Systemmanagement.**

- **Tivoli Event Pump für z/OS ermöglicht die automatische Erfassung und Weiterleitung von Zustands-und Statusereignissen von z/OS-Rechnern und Subsystemen, einschließlich CICS, IMS, DB2 sowie Drittanbieterprodukten.**

- **Tivoli Advanced Backup und Recovery für z/OS stellt eine zuverlässige und genaue Sicherung und Wiederherstellung von Anwendungsdaten auf z/OS-Rechnern bereit, und verfolgt Sicherungen für Prüfprotokolle.**

- **Tivoli OMEGAMON XE für z/OS bietet detailliertes Überwachungs- und Problem-Management für IBM System z und IBM zEnterprise-Systeme. Hiermit kann die Sichtbarkeit, Benutzerfreundlichkeit und Leistung verbessert werden.**

- **Tivoli System Automation für z/OS ist eine Policy-basierte, selbstheilende und hochverfügbare Lösung, um die Effizienz und Verfügbarkeit von kritischen Systemen und Anwendungen zu maximieren.**

- **Mit Tivoli Workload Scheduler für z/OS können Sie die Verarbeitung von IBM System z Workloads automatisieren, planen und steuern. Es fungiert als eine virtueller Kontrollpunkt, der für unternehmensweite Produktion-Workloads die Geschwindigkeit zu maximieren und Ressourcen zu optimieren ermöglicht.**

Jedes dieser Produkte braucht Experten-Wissen für den Einsatz und eine umfangreiche Einarbeitungszeit.

Während IBM Für Software Produkte wie CICS, VSAM und MQSeries ein de Facto Monopol besitzt, gilt das nicht für die Tivoli Produkte (und andere Software wie z.B. RACF). Eine Reihe von anderen Software Unternehmen (Independent Software Vendor, ISV) wie Computer Accociates, BMC und Hewlett Packard bieten viele vergleichbare Software Produkte an, die als Alternative zu den Tivoli Produkten eingesetzt werden können.