

Approximative analytische Leistungsbewertung am Beispiel eines UNIX-basierten Multiprozessor Betriebssystems

Gunter Bolch
Universität Erlangen-Nürnberg
IMMD IV
Martensstrasse 1
D – 91058 Erlangen
bolch@informatik.uni-erlangen.de

Stefan Greiner
Duke University
Dept. of Electrical Engineering
Durham, NC 27708-0291, USA
Box 90291
greiner@ee.duke.edu

Zusammenfassung

Ausgangspunkt der vorliegenden Arbeit sind die am Institut für Mathematische Maschinen und Datenverarbeitung (IV) entwickelten Modelle [7] für drei Varianten eines auf UNIX basierenden Multiprozessorbetriebssystems. Diese Modelle wurden ursprünglich mittels Markovmethoden und diskreter ereignisorientierter Simulation analysiert, um Leistungsparameter zu erhalten, wobei Messungen am realen System zur Verfügung standen. In diesen Modellen ist Klassenwechsel erlaubt, die Abarbeitung der Aufträge erfolgt nach Prioritäten, es liegt eine gemischte Prioritätsstrategie vor, einzelne Auftragsklassen dürfen alle Knoten besuchen (symmetrischer Fall) andere Auftragsklassen dagegen nur bestimmte Knoten (asymmetrischer Fall). Um den hohen zeitlichen Analyseaufwand zu reduzieren (ca. zwei Tage für Simulation und ca. 25 Minuten für Markovanalyse), wurden die Modelle so transformiert, daß schnelle Standardverfahren wie z.B. die Mittelwertanalyse angewendet werden können. Mittels dieser neuen Technik wurde die Rechenzeit auf weniger als eine Sekunde reduziert und die Ergebnisse, die wir erhalten haben, lagen sehr nahe an den gemessenen Werten.

Approximate analytical performance evaluation of a UNIX-based multiprocessor operating system

Abstract

The motivation of this paper is based on models for three variants of a UNIX based multiprocessor operating system developed at the Institute for mathematical machines and data processing IV [7]. These models were analyzed by discrete event simulation and Markovian methods and were compared against measurements from the real system. In this models class switching is allowed, jobs have priorities with a mixed priority strategy (HOL-PR), some job classes can visit each node while other jobs are only allowed to visit special nodes. To reduce the time to analyze the system (about two days with discrete event simulation, about 25 minutes with Markovian analysis), we developed a new approximate technique based on the well known Mean Value Analysis (MVA). With this new technique the computation time was reduced to less than 1 second.

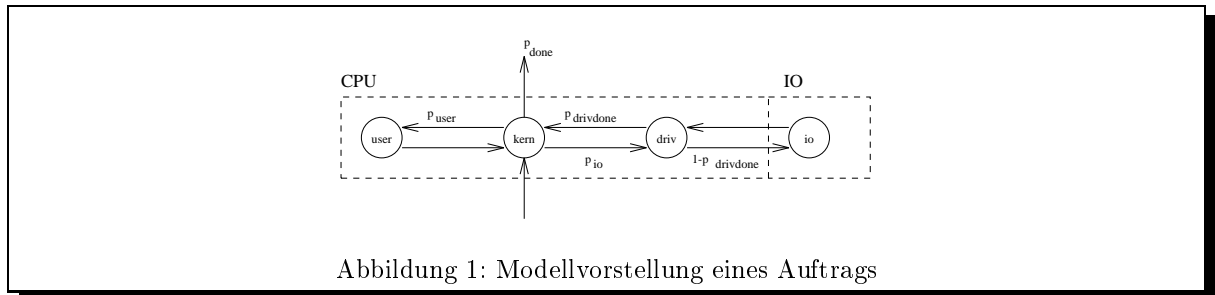
Schlüsselwörter : Leistungsbewertung, Warteschlangenmodelle, UNIX, Multiprozessorsystem, Mittelwertanalyse, gemischte Prioritäten, Shadow-Technik, Klassenwechsel, heterogene Knoten.

1 Einführung

Leistungsbewertung von Rechen- und Betriebssystemen ist aufgrund der Komplexität heutiger Systeme von großem Interesse. Ziel der Leistungsbewertung ist hierbei das Aufdecken von Engpässen und damit verbunden die Optimierung des Systems bzgl. verschiedener Parameter wie z.B. Durchsatz oder Verweilzeit. Da Leistungsbewertung von Rechen- und Betriebssystemen Hand in Hand mit deren Entwurf und Entwicklung

gehen sollte, steht ein reales System häufig nicht zur Verfügung und Meßmethoden sind daher nicht möglich. Zur Leistungsbewertung eines in der Planung befindlichen Systems muß man sich daher ein geeignetes (mathematisches) Modell des Systems verschaffen, wobei Warteschlangenmodelle wegen ihrer Anschaulichkeit besonders geeignet sind. Motivation für die vorliegende Arbeit waren die am Institut für mathematische Maschinen und Datenverarbeitung (IV) der Friedrich-Alexander-Universität Erlangen-Nürnberg im Rahmen einer Dissertation entwickelten Modelle für mehrere Varianten eines auf UNIX basierenden Multiprozessorbetriebssystems [7].

1.1 Modellvorstellung eines Auftrags



In Bild 1 ist die Modellvorstellung eines Auftrags wiedergegeben. Ein Auftrag beginnt seinen Lebenslauf immer im Kernkontext kern und wechselt dann mit den entsprechenden Übergangswahrscheinlichkeiten in den Benutzerkontext user und Treiberkontext driv. Ein Verlassen des Systems ist für einen Auftrag nur vom Kernkontext aus möglich.

1.2 Modellvorstellung des Rechen- und Betriebssystems

Das System wird durch ein Wartnetz modelliert mit folgenden Annahmen:

- das Netz ist geschlossen und es befinden sich genau K Aufträge im System.
- die Bedienzeiten sind exponentiell verteilt.
- die peripheren Geräte werden in Form von drei Platten modelliert, die jeweils mit der Wahrscheinlichkeit $\frac{1}{3}$ belegt werden und daher als lastabhängige Bedienstation modelliert werden. s_{io} stellt hierbei die mittlere Bedienzeit eines Auftrags in msec an der lastabhängigen Bedienstation.

$s_{io}(1) = 28.00$	$s_{io}(6) = 12.444$
$s_{io}(2) = 18.667$	$s_{io}(7) = 12.000$
$s_{io}(3) = 15.555$	$s_{io}(8) = 11.667$
$s_{io}(4) = 14.000$	$s_{io}(9) = 11.407$
$s_{io}(5) = 13.067$	$s_{io}(10) = 11.200$

Tabelle 1: Bedienzeiten der lastabhängigen Bedienstation

- das System besitzt drei Kontexte (Klassen) user, kern, driv mit der Prioritätsreihenfolge driv > kern > user.
- Aufträge im Kontext user können jederzeit durch Aufträge im Kontext driv oder kern verdrängt werden. Weitere Verdrängung sind im System nicht möglich.

- Klassenwechsel ist möglich.
- alle Einschwingvorgänge sind abgeklungen.
- je nach Modell können im System Hilfsprozessoren (APU's = Associate Processing Units) verwendet werden.
- als Standardparametersatz wird gewählt

$K = 10$	$\#Apu's = 2$
$p_{io} = 0.05$	$s_{user} = 0.25...20.0$
$p_{done} = 0.01/0.005$	$s_{kern} = 1.0$
$p_{drivdone} = 0.4$	$s_{driv} = 0.5$

Tabelle 2: Standardparametersatz

- ein Zustand wird beschrieben wird durch ein Paar

(Knotennummer, Klassennummer)

Knotennummer 1 : CPU
 Knotennummer 2 : IO
 Klassennummer 1 : driv
 Klassennummer 2 : kern
 Klassennummer 3 : user

wobei für die Zustandsübergänge folgende Wahrscheinlichkeiten angenommen werden:

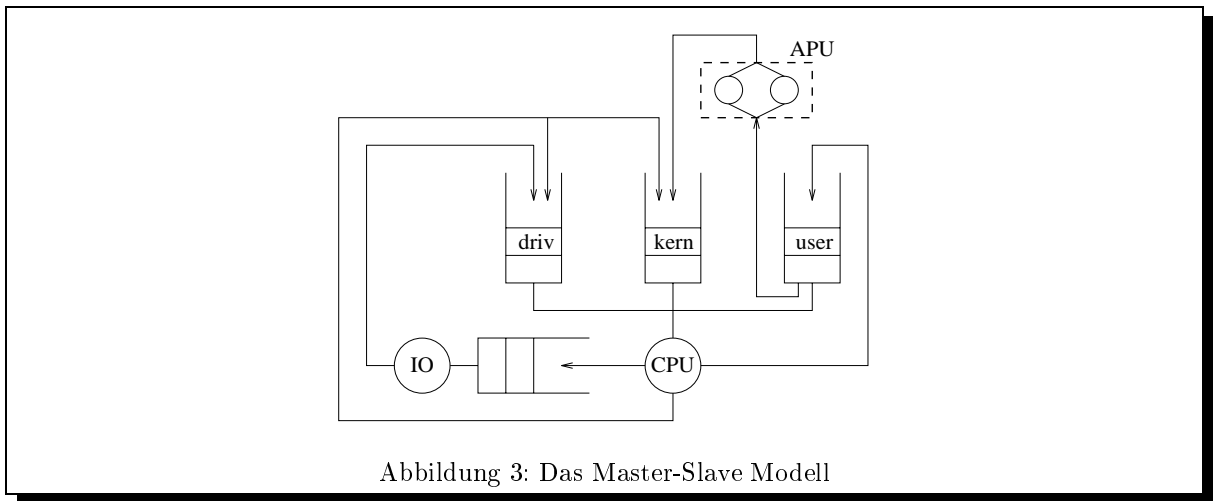
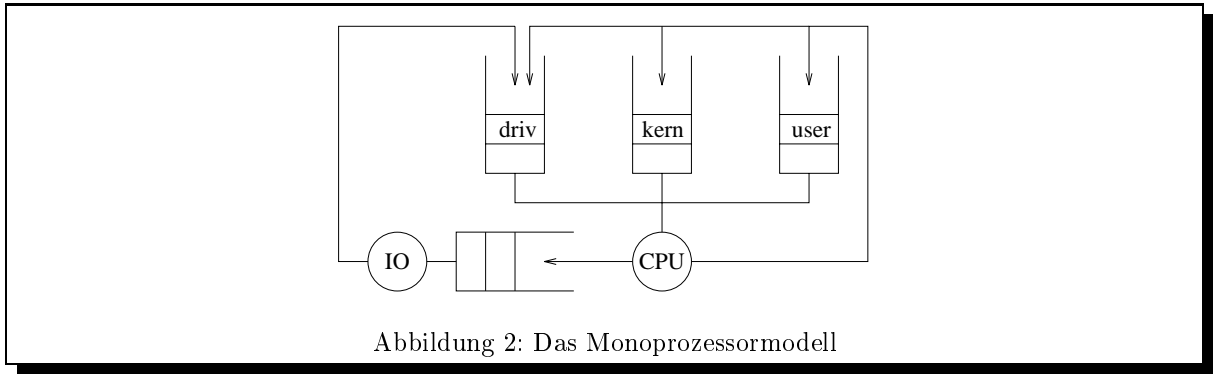
	(1,1)	(1,2)	(1,3)	(2,1)
(1,1)	0	$p_{drivdone}$	0	$1 - p_{drivdone}$
(1,2)	p_{io}	p_{done}	p_{user}	0
(1,3)	0	1	0	0
(2,1)	1	0	0	0

Tabelle 3: Übergangsmatrix des UNIX-Modells

Im folgenden werden die einzelnen Betriebssystemmodelle als Warteschlangennetze vorgestellt. Grundmodell ist dabei das Monoprozessormodell. Eine Erweiterung dieses Modells ist das Master-Slave Modell, wobei bei diesem Modell Hilfsprozessoren (APU's) vorhanden sind, die Aufträge im Userkontext (user) bearbeiten können. Falls auf den Hilfsprozessoren zusätzlich auch noch Aufträge im Kernkontext (kern) bearbeitet werden können, erhält man schließlich das sog. Associated-Processor Modell.

1.3 Das Monoprozessormodell

Das Modell in Bild 2 ist das sog. Monoprozessormodell. Aus abstrakter Sicht hat man ein geschlossenes Warteschlangennetz mit drei Auftragsklassen. An der CPU liegt eine Kombination aus verdrängender- und nichtverdrängender Bedienstrategie vor. D.h. ein Auftrag im Userkontext (user) kann jederzeit durch einen Auftrag im Kernkontext (kern) oder Treiberkontext (driv) verdrängt werden. Andere Verdrängungen sind im System nicht möglich. Ist die CPU frei und stehen sowohl kern als auch driv Aufträge zur Abarbeitung an, so haben driv Aufträge höhere Priorität. Eine Bedingung für dieses Systemmodell ist, daß zu jedem Zeitpunkt höchstens ein Job in der CPU aktiv sein kann.



1.4 Das Master-Slave Modell

Für das Master-Slave Modell (Bild 3) gelten die gleichen Annahmen wie im Monoprozessormodell, wobei noch zwei sog. APU's (Hilfsprozessoren) hinzukommen. Diese APU's können nur Aufträge im Kontext user bearbeiten. Wenn sowohl CPU als auch APU frei sind und ein Job zur Abarbeitung ansteht, so wird zufällig entschieden, auf welchem Prozessor (CPU oder APU) er bearbeitet wird.

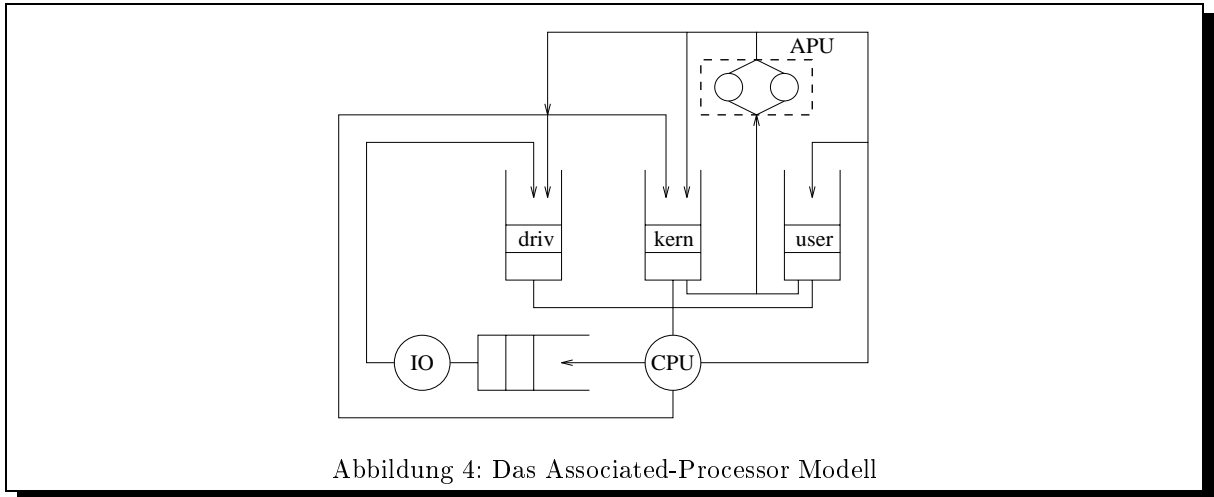
1.5 Das Associated-Processor Modell

Die Annahmen für dieses Modell (Bild 4) sind die gleichen wie beim Master Slave Modell, aber nun können auf der APU noch zusätzlich Jobs im Kontext kern bearbeitet werden. Da kern Jobs eine höhere Priorität haben als user Jobs, ist nun auch auf den APU's eine Verdrängung möglich. User Aufträge können jederzeit durch kern-und driv Aufträge verdrängt werden, kern-und driv-Aufträge können sich dagegen nicht gegenseitig verdrängen.

1.6 Probleme bei den Modellen

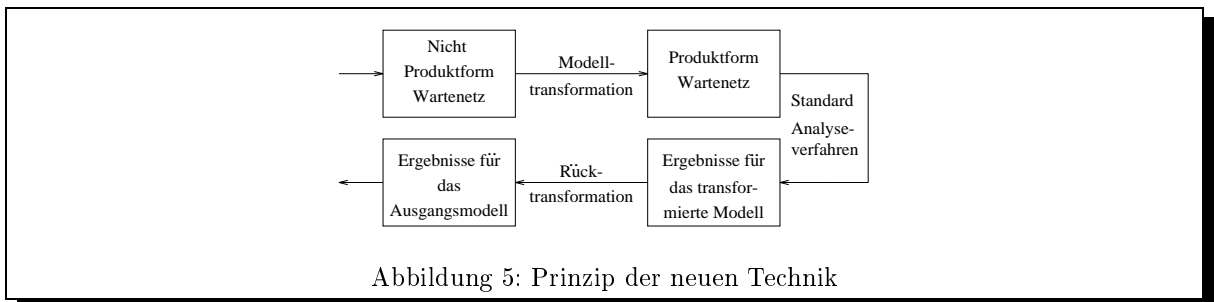
Wegen der folgenden Annahmen können Standard-Analyseverfahren nicht angewendet werden:

- die Abarbeitung der Aufträge erfolgt nach Prioritäten.
- es liegt eine gemischte Prioritätsstrategie vor, d.h. einige Auftragsklassen können verdrängt werden, andere Auftragsklassen dagegen nicht.



- Klassenwechsel ist erlaubt.
- bestimmte Auftragsklassen können jeden Knoten besuchen, andere Auftragsklassen dagegen nur bestimmte Knoten (heterogene Knoten).

Für einen Teil dieser Probleme existieren in der Literatur bereits separate Lösungen. Ziel war es jedoch, diese Lösungen zu einem geeigneten Verfahren zu kombinieren und zu erweitern. Das Prinzip der neuen Technik wird in Bild 5 gezeigt:



2 Leistungsgrößen von Warteschlangennetzen

Für Warteschlangennetze sind folgende allgemeine Bezeichnungen üblich [2] :

- N : Anzahl der Knoten im Netz.
- R : Anzahl der Auftragsklassen.
- \underline{k} : Populationsvektor (k_1, k_2, \dots, k_R) , wobei k_i die Anzahl der Aufträge in Klasse i angibt ($1 \leq i \leq R$).

K	:	Anzahl der Aufträge im geschlossenen Netz mit $K = \sum_{i=1}^R k_i$.
$(\underline{k} - 1_r)$:	Populationsvektor bei einem Auftrag weniger in Klasse r .
μ_{ir}	:	mittlere Bedienrate eines Auftrags der Klasse r am Knoten i .
$s_{i,r} = \frac{1}{\mu_{ir}}$:	mittlere Bedienzeit eines Auftrags der Klasse r am Knoten i .
λ_{ir}	:	mittlere Ankunftsrate eines Auftrags der Klasse r am Knoten i .
m_i	:	Anzahl der Bedieneinheiten am Knoten i .
$p_{ir,js}$:	Wahrscheinlichkeit, daß ein Auftrag der Klasse r nach Bedienung am Knoten i in die Klasse s des Knoten j wechselt.
e_{ir}	:	Besuchshäufigkeit eines Auftrags der Klasse r am Knoten i .

$$e_{ir} = \sum_{j=1}^N \sum_{s=1}^R e_{js} \cdot p_{js,ir}$$

ρ_{ir} : Auslastung des Knoten i durch Aufträge der Klasse r .

$$\rho_{ir} = \frac{\lambda_{ir}}{m_i \cdot \mu_i}$$

\bar{w}_{ir} : mittlere Wartezeit eines Auftrags der Klasse r am Knoten i .
 $\bar{t}_{ir}(\underline{k})$: mittlere Antwortzeit (Verweilzeit) eines Auftrags der Klasse r am Knoten i bei einer gegebenen Auftragspopulation \underline{k} im Netz.

$$\bar{t}_{ir}(\underline{k}) = \bar{w}_{ir} + \frac{1}{\mu_{ir}}$$

\bar{q}_{ir} : mittlere Warteschlangenlänge von Aufträgen der Klasse r im Knoten i
 $\bar{k}_{ir}(\underline{k})$: mittlere Anzahl von Aufträgen der Klasse r im Knoten i bei einer gegebenen Auftragspopulation \underline{k} im Netz.

3 Warteschlangennetze ohne Klassenwechsel

3.1 Beschreibung

Ist in einem Warteschlangennetz kein Klassenwechsel möglich, so ist die Anzahl der Aufträge innerhalb jeder Klasse konstant. Eine Klasse ist hierbei eine Menge von Aufträgen mit gleichem Anforderungsprofil. Für diesen Typ von Netzen gibt es viele exakte und auch approximative Analysemethoden. Viele dieser Verfahren sind in dem am Institut für mathematische Maschinen und Datenverarbeitung (IV) entwickelten Programmpaket PEPSY-QNS (**P**erformance **E**valuation and **P**rediction **S**ystem for **Q**ueueing **N**etwork**S**) [2, 9] implementiert.

Da die spätere Analyse der vorgestellten Modelle mittels einer geeignet modifizierten Mittelwertanalyse durchgeführt werden sollen, werden im folgenden die Mittelwertanalyse sowie die entsprechenden Modifikationen schrittweise eingeführt.

3.2 Die Mittelwertanalyse

Die Mittelwertanalyse [14] dient zur exakten Analyse von Netzen mit folgenden Knotentypen (sog. Produktformknoten):

- Typ 1 : M / M / m - FCFS
- Typ 2 : M / G / 1 - PS (RR)
- Typ 3 : M / G / ∞ (Infinite Server)

- Typ 4 : M / G / 1 - LCFS PR

Die beiden folgenden Beziehungen bilden die Grundlage für dieses Analyseverfahren:

- Gesetz von Little: $\bar{k} = \lambda \cdot \bar{t}$
- Theorem über die Verteilung beim Ankunftszeitpunkt (sog. Ankunftstheorem) von Reiser und Lavenberg :

$$\bar{t}_i(K) = \frac{1}{\mu_i} \cdot (1 + \bar{k}_i(K - 1))$$

Aus diesen beiden Beziehungen ergibt sich durch geeignete Modifikation die für die Mittelwertanalyse zentrale Formel, auf die sich im weiteren alle Änderungen beziehen:

$$\bar{t}_{ir}(\underline{k}) = \begin{cases} \frac{1}{\mu_{ir}} \cdot \left(1 + \sum_{s=1}^R \bar{k}_{is}(\underline{k} - 1_r) \right) & \text{Typ-1,2,4} \\ & (m_i = 1) \\ \frac{1}{\mu_{ir} \cdot m_i} \cdot \left(1 + \sum_{s=1}^R \bar{k}_{is}(\underline{k} - 1_r) + \right. \\ \quad \left. \sum_{j=0}^{m_i-2} (m_i - j - 1) \cdot p_i(j | \underline{k} - 1_r) \right) & \text{Typ-1} \\ & (m_i > 1) \\ \frac{1}{\mu_{ir}} & \text{Typ-3} \end{cases}$$

3.3 Prioritäten

Wird eine Prioritätsstrategie angewendet, so verletzt dies im allgemeinen die Produktformbedingung (eine Ausnahme bilden Produktformnetze mit Typ-4 Knoten). Es ist daher nur noch eine approximative Lösung möglich. Im weiteren wird angenommen, daß die einzelnen Auftragsklassen (Prioritätsklassen) linear geordnet sind, wobei Klasse 1 die höchste Priorität besitzt. Bei der Herleitung wird zunächst nur ein einzelner Knoten betrachtet. Die Ergebnisse werden dann in das Gesamtnetz (mittels der Mittelwertanalyse) integriert.

3.3.1 M/M/1-FCFS PR (preemptive resume)

In diesem Fall verdrängt ein ankommender Auftrag höherer Priorität einen gerade in Bedienung befindlichen Auftrag mit niedrigerer Priorität. D.h. ein Auftrag höherer Priorität kann bei Ankunft im System sofort bedient werden, wenn kein anderer Auftrag höherer Priorität im System ist. Wenn alle Aufträge höherer Priorität bedient sind, wird der verdrängte Auftrag an der Stelle weiterbearbeitet, an der er unterbrochen wurde. Die mittlere Antwortzeit eines Knotens setzt sich in diesem Fall zusammen aus

- der Bedienzeit

$$\frac{1}{\mu_r}$$

- der Zeit, die gewartet werden muß, bis alle Aufträge höherer oder gleicher Priorität im Knoten abgearbeitet sind

$$\sum_{s=1}^r \frac{\bar{k}_s}{\mu_s}$$

- der Zeit für die Bearbeitung von Aufträgen höherer Priorität, die während der Antwortzeit \bar{t}_r eintreffen

$$\sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s}$$

Damit ergibt sich für die Prioritätsklasse r ($1 \leq r \leq R$) [2]

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s}{\mu_s} + \sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

Durch Ausklammern von \bar{t}_r und Anwenden des Theorems über die Verteilung beim Ankunftszeitpunkt erhält man für den Knoten i [1] :

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

3.3.2 M/M/1-FCFS HOL (nonpreemptive Head of Line):

Ein Auftrag höherer Priorität muß warten, bis ein in Bedienung befindlicher Auftrag fertig bearbeitet ist. Die mittlere Antwortzeit setzt sich somit zusammen aus

- der eigenen Bedienzeit

$$\frac{1}{\mu_r}$$

- der Verzögerung durch Aufträge höherer Priorität, die während der Wartezeit $(\bar{t}_r - \frac{1}{\mu_r})$ eintreffen

$$\sum_{s=1}^{r-1} \left(\bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s}$$

- der Verzögerung des noch in Bearbeitung befindlichen Auftrags

$$\sum_{s=1}^R \frac{\rho_s}{\mu_s}$$

- der Bedienzeit für Aufträge gleicher oder höherer Priorität, die sich bereits in der Warteschlange befinden

$$\sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s}$$

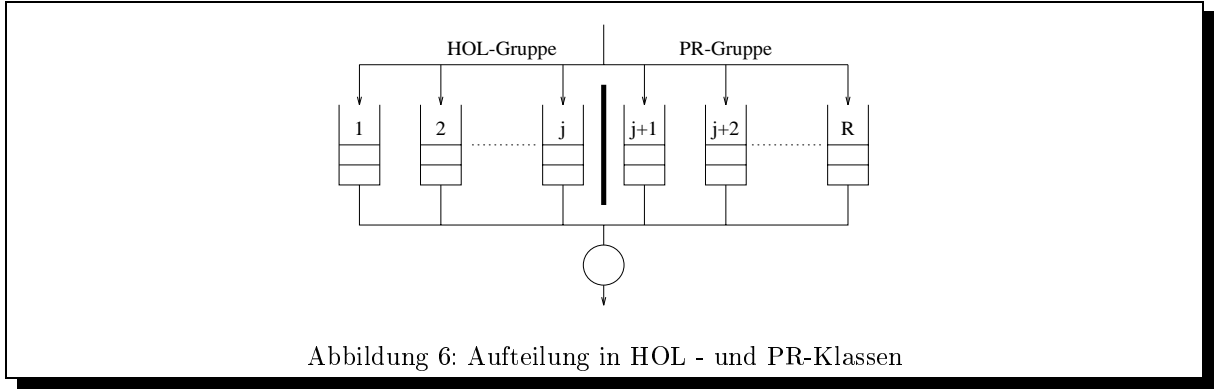
Damit ergibt sich für die Prioritätsklasse r ($1 \leq r \leq R$)

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s} + \sum_{s=1}^R \frac{\rho_s}{\mu_s} + \sum_{s=1}^{r-1} \left(\bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

Durch Ausklammern von \bar{t}_r und Anwenden des Theorems über die Verteilung beim Ankunftszeitpunkt erhält man für den Knoten i [1] :

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}} + \sum_{s=r+1}^R \frac{\rho_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

Wird diese Formel anstelle des ursprünglichen \bar{t}_{ir} in der Mittelwertanalyse verwendet, so erhält man ein approximatives Verfahren zur Analyse von Prioritätsnetzen mit PR oder HOL Strategie. Diese Formeln liefern für Auslastungen bis $\rho = 0.7$ recht gute Ergebnisse. Für höhere Auslastungen nimmt jedoch die Genauigkeit bei den Klassen $r > 1$ ab. Die Ergebnisse für Klasse 1 Aufträge sind dagegen exakt.



3.3.3 Kombination von PR und HOL

Bisher haben wir den Fall betrachtet, daß nur eine reine HOL-oder PR-Strategie möglich war. Wir wollen nun auch den Fall betrachten, daß eine gemischte Prioritätsstrategie (Bild 6) möglich ist. In diesem Fall wird ein Teil der Aufträge nach HOL abgearbeitet, ein anderer nach PR. Aufträge aus der HOL-Gruppe haben grundsätzlich eine höhere Priorität als Aufträge aus der PR-Gruppe (Bild 6).

Man teilt die Aufträge in die beiden Gruppen HOL und PR auf, wobei gilt :

- Ein ankommender Auftrag aus der HOL-Gruppe kann einen gerade in Bearbeitung befindlichen Auftrag aus der HOL-Gruppe nicht verdrängen. Dagegen kann ein Auftrag aus der PR-Gruppe jederzeit von einem HOL-Auftrag verdrängt werden. Damit wird die Wartezeit eines HOL-Auftrags nicht beeinflusst durch die Ankunft von PR-Aufträgen.

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s - \rho_s}{\mu_s} + \sum_{s=1}^j \frac{\rho_s}{\mu_s} + \sum_{s=1}^{r-1} \left(\bar{t}_r - \frac{1}{\mu_r} \right) \cdot \frac{\lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

Durch Ausklammern von \bar{t}_r und Anwenden des Theorems über die Verteilung beim Ankunftszeitpunkt erhält man für den Knoten i :

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}} + \sum_{s=r+1}^j \frac{\rho_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

- Ein PR-Auftrag kann jederzeit durch einen PR-Auftrag höherer Priorität oder durch einen beliebigen HOL-Auftrag verdrängt werden. D.h. ein PR-Auftrag wird erst bearbeitet, wenn kein PR-Auftrag höherer Priorität und kein HOL-Auftrag zur Abarbeitung ansteht. Damit ergibt sich für die Prioritätsklasse r :

$$\bar{t}_r = \sum_{s=1}^r \frac{\bar{k}_s}{\mu_s} + \sum_{s=1}^{r-1} \frac{\bar{t}_r \cdot \lambda_s}{\mu_s} + \frac{1}{\mu_r}$$

Durch Ausklammern von \bar{t}_r und Anwenden des Theorems über die Verteilung beim Ankunftszeitpunkt erhält man für den Knoten i :

$$\begin{aligned} \bar{t}_{ir}(\underline{k}) &= \frac{1}{\mu_{ir}} + \frac{\sum_{s=1}^r \frac{\bar{k}_{is}(\underline{k}-1_r)}{\mu_{is}}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{is}} \\ \tilde{\rho}_{is} &= \rho_{is}(\underline{k} - \bar{k}_{is}) \end{aligned}$$

Werden diese Formeln anstelle von \bar{t}_{ir} in die ursprüngliche Mittelwertanalyse eingesetzt, so erhält man eine approximierte Mittelwertanalyse zur Analyse von geschlossenen Netzen mit gemischter Prioritätsstrategie.

4 Warteschlangennetze mit Klassenwechsel

4.1 Beschreibung

Die Menge aller möglichen Übergänge zwischen verschiedenen Knoten und Klassen in einem Warteschlangennetzwerk kann in Form einer Übergangsmatrix dargestellt werden. Dies bedeutet aber insbesondere, daß durch den Klassenwechsel die Anzahl der Aufträge innerhalb einer Klasse nicht mehr konstant ist.

Nach [3] ist ein zulässiger Zustand in einem Warteschlangenmodell ohne Klassenwechsel durch folgende vier Bedingungen charakterisiert :

1. die Anzahl der Aufträge jeder Klasse an einer Station ist nicht negativ, d.h.

$$k_{ir} \geq 0 \quad \forall 1 \leq r \leq R \quad \wedge \quad 1 \leq i \leq N$$

2. für die Aufträge in einer Station gilt:

$$k_{ir} > 0 \quad \iff$$

mit Wahrscheinlichkeit größer null betritt ein Auftrag der Klasse r den Knoten i .

3. die Gesamtzahl der Aufträge in den Knoten ist

$$K = \sum_{r=1}^R \sum_{i=1}^N k_{ir}.$$

4. die Zahl der Klasse r Aufträge im Netz ist konstant.

$$k_r = \sum_{i=1}^N k_{ir} = \text{const.} \quad \forall 1 \leq r \leq R$$

Ist Klassenwechsel möglich, so werden die Bedingungen 1-3 zwar erfüllt, Bedingung 4 ist aber nicht erfüllbar, da die Anzahl der Aufträge innerhalb einer Klasse nicht mehr konstant ist, sondern abhängig vom Beobachtungszeitpunkt einen Wert $x \in \{0..K\}$ annimmt. Um diesen Mißstand zu beheben, wird das Konzept der Ketten eingeführt.

4.2 Das Kettenkonzept

4.2.1 Die Idee der Kettenzerlegung

Sei P eine allgemeine Übergangsmatrix mit endlichem Zustandsraum Ω . Dieser Zustandsraum läßt sich zerlegen in disjunkte Mengen

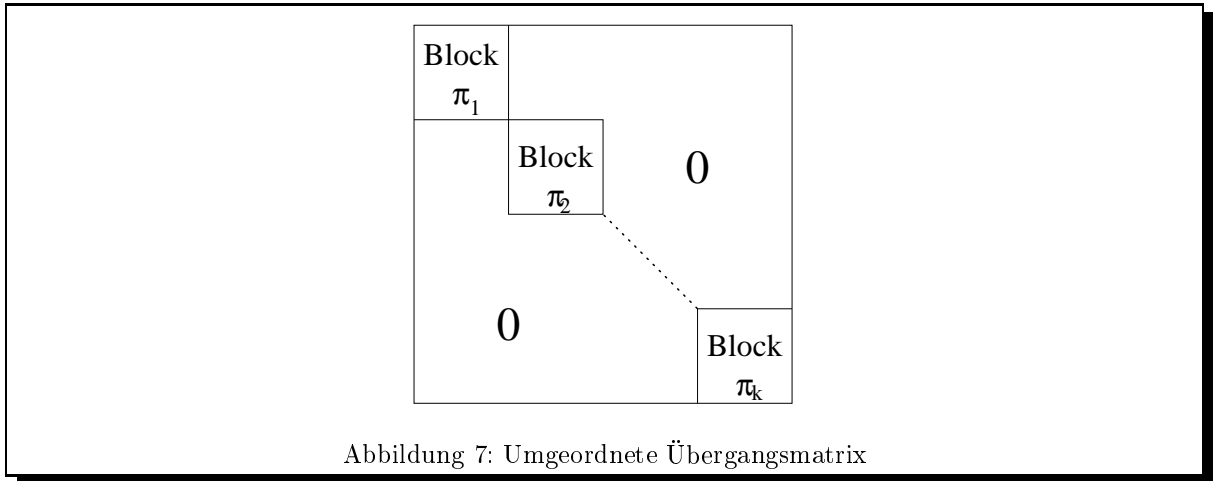
$$\Omega = \pi_1 + \pi_2 + \dots + \pi_k$$

wobei π_i die Menge der rekurrenten Zustände ist (d.h. die mittlere Wiederkehrzeit in diese Zustände ist endlich).

Durch geeignete Umordnung der Blöcke π_i erhält man dann die Übergangsmatrix von Bild 7. Die einzelnen Teilketten π_i sind hierbei disjunkt, ergodisch und bilden selbst wieder eine Übergangsmatrix. Da die einzelnen Teilketten π_i disjunkt sind, ist ein Wechsel zwischen den einzelnen Ketten nicht möglich. Startet man also in einem rekurrenten Block, so verläßt man diesen nie.

Durch diese Zerlegung in Teilketten wird erreicht, daß die Anzahl der Aufträge in einer rekurrenten Teilkette immer konstant bleibt, womit dann auch Bedingung 4 für Warteschlangennetze erfüllt wird.

Muntz [13] zeigte, daß ein geschlossenes Netz mit U ergodischen Teilketten äquivalent ist zu einem geschlossenen Netz mit U Auftragsklassen. Ist kein Klassenwechsel möglich, so ist die Anzahl der Ketten gleich der Anzahl der Klassen. Ist dagegen Klassenwechsel erlaubt, so ist die Zahl der Ketten echt kleiner als die Zahl der Klassen.



4.2.2 Bestimmung der Teilketten

Durch die Übergangsmatrix $P = [p_{ir,js}]$ wird eine zeitdiskrete Markovkette definiert, deren Zustände Paare der Form (i, r) sind mit

- i : Knotennummer
- r : Klassennummer

1. Man definiert R Mengen, die folgender Eigenschaft genügen

$$E_r = \{ s : \text{der Zustand } (j, s) \text{ kann in } \geq 0 \text{ Schritten von einem Zustand } (i, r) \text{ aus erreicht werden} \},$$

$$\forall 1 \leq r \leq R \quad \wedge \quad 1 \leq s \leq R$$

2. Elimination von gleichen Mengen und Untermengen, so daß

$$U \leq R \text{ Mengen } \pi_1 \dots \pi_U$$

übrigbleiben.

3. Bestimmung der Anzahl der Aufträge in jeder Kette.

$$K'_s = \sum_{r \in \pi_s} K_r$$

$$\forall 1 \leq s \leq U$$

4.2.3 Beispiel

Gegeben sei folgende Übergangsmatrix

P	1,1	1,2	1,3	2,1	2,2	2,3	3,1	3,2	3,3
1,1	0	0.4	0	0	0.3	0	0	0.3	0
1,2	0.3	0	0	0	0	0	0	0.7	0
1,3	0	0	0	0	0	0.3	0	0	0.7
2,1	0	0	0	0	1	0	0	0	0
2,2	0.3	0	0	0	0	0	0.7	0	0
2,3	0	0	0.3	0	0	0	0	0	0.7
3,1	1.0	0	0	0	0	0	0	0	0
3,2	0	0	0	1.0	0	0	0	0	0
3,3	0	0	0.4	0	0	0.6	0	0	0

Tabelle 4: Beispiel einer Übergangsmatrix

Die Zerlegung in Ketten liefert

$$\begin{aligned} E_1 &= \{1, 2\} \\ E_2 &= \{1, 2\} \\ E_3 &= \{3\} \end{aligned}$$

Durch Elimination der Untermengen und identischen Mengen erhält man die beiden Ketten

$$\begin{aligned} \pi_1 &= \{1, 2\} \\ \pi_2 &= \{3\} \end{aligned}$$

Die kettenzerlegte Matrix erhält somit die Form

P	1,1	1,2	2,1	2,2	3,1	3,2	1,3	2,3	3,3
1,1	0	0.4	0	0.3		0.3	0	0	0
1,2	0.3	0	0	0	0	0.7	0	0	0
2,1	0	0	0	1.0	0	0	0	0	0
2,2	0.3	0	0	0	0.7	0	0	0	0
3,1	1.0	0	0	0	0	0	0	0	0
3,2	0	0	1.0	0	0	0	0	0	0
(1,3)	0	0	0	0	0	0	0	0.3	0.7
(2,3)	0	0	0	0	0	0	0.3	0	0.7
(3,3)	0	0	0	0	0	0	0.4	0.6	0

Tabelle 5: Umgeordnete Übergangsmatrix

Startet man in einem der beiden Blöcke π_1 oder π_2 , so kann man diesen nicht mehr verlassen, da die Übergangswahrscheinlichkeiten zwischen Zuständen in verschiedenen Blöcken Null sind.

4.3 Erweiterung der Mittelwertanalyse auf Klassenwechsel

Die Erweiterung der Mittelwertanalyse auf Netze mit Klassenwechsel ist mittels des Kettenkonzeptes sehr leicht möglich und resultiert in einem zur ursprünglichen Mittelwertanalyse isomorphen Verfahren [3]. Durch den Übergang von Klassen zu Ketten ist es aber notwendig, Klassengrößen in Kettengrößen umzuwandeln.

- Besuchshäufigkeiten

Innerhalb einer Kette kann ein Auftrag verschiedene Zustände (i, j)

i = Knotennummer

j = Klassennummer

erreichen. Damit erhält man die Besuchshäufigkeit in der Kette durch Summation über die Besuchshäufigkeiten in den einzelnen zur Kette gehörigen Klassen.

$$e_{iq}^* = \frac{\sum_{r \in \pi_q} e_{ir}}{\sum_{r \in \pi_q} e_{1r}}$$

wobei e_{i1}^* auf 1 normiert wird.

- Die Anzahl der Aufträge in einer Kette ist zwar konstant, jedoch können die Aufträge innerhalb einer Kette ihre Klassenzugehörigkeit wechseln. Besucht nun ein Auftrag der Kette q den Knoten i , so läßt sich nicht feststellen, um welche Klasse es sich handelt, da eine Kette als ein homogenes Gebilde betrachtet wird, bei dem durch den Übergang

$$\text{Klasse} \quad \Longrightarrow \quad \text{Kette}$$

die Information über die Klassenzugehörigkeit verlorengeht.

Da aber i.a. an einem Knoten Aufträge verschiedener Klassen verschiedene Bedienzeiten besitzen,

benötigt man Informationen über die Klassenzugehörigkeit. Diese (fehlende) Information wird ersetzt durch den Skalierungsfaktor α_{ir} ($r \in \pi_q$)

$$\alpha_{ir} = \frac{e_{ir}}{\sum_{s \in \pi_q} e_{is}}$$

Dieser Skalierungsfaktor gibt den Anteil der Klasse r Aufträge an, die Knoten i besuchen.

- Bedienzeit

Da eine Kette aus $n \geq 1$ Auftragsklassen besteht, erhält man die Kettenbedienzeit als Summe über die Klassenbedienzeiten gewichtet mit dem Skalierungsfaktor α .

Die Gewichtung dient dazu, die fehlende Information über die Klassenzugehörigkeit auszugleichen.

$$s_{iq} = \frac{1}{\mu_{iq}} = \sum_{r \in \pi_q} s_{ir} \cdot \alpha_{ir}$$

Damit erhält man folgenden Algorithmus zur Berechnung der Leistungsgrößen :

1. bestimme die Besuchshäufigkeiten e_{ir} im Ausgangsnetz.
2. zerlege die Übergangsmatrix P in Ketten $\pi_1 \dots \pi_U$ und bestimme die Auftragszahl k_i^* in den einzelnen Ketten.
3. berechne die kettenbezogenen Besuchshäufigkeiten e_{iq}^* .
4. berechne die Skalierungsfaktoren α_{ir} .
5. berechne die kettenbezogenen Bedienzeiten s_{iq} .
6. berechne die kettenbezogenen Leistungsgrößen mittels der erweiterten Mittelwertanalyse, wobei die Ketten so behandelt werden wie Klassen in einem System ohne Klassenwechsel
7. transformiere die kettenbezogenen Leistungsgrößen in klassenbezogene Leistungsgrößen zurück.

$$\begin{aligned} \bar{t}_{ir}(\underline{k}_u) &= s_{ir} \cdot (1 + k_i^*(\underline{k}_u - 1_q)) \\ \lambda_{ir}(\underline{k}_u) &= \alpha_{ir} \cdot \lambda_{iq}^*(\underline{k}_u) \quad r \in \pi_q \\ \rho_{ir}(\underline{k}_u) &= s_{ir} \cdot \lambda_{ir}(k_u) \end{aligned}$$

Folgende Punkte sind zu beachten :

- die Dimension des Populationsvektors ist $U \leq R$.
- im Fall $U = R$ liegt kein Klassenwechsel vor.
- wie man aus obiger Darstellung erkennt, müssen bereits bestehende Algorithmen zur Mittelwertanalyse nicht geändert sondern nur um folgende Punkte erweitert werden :
 - bestimme die Anzahl der Ketten.
Ist die Anzahl der Ketten gleich der Anzahl der Klassen, so ist kein Klassenwechsel möglich und man kann die Mittelwertanalyse mit den Originalparametern ausführen.
Ist die Anzahl der Ketten echt kleiner als die Anzahl der Klassen, so ist im Netz Klassenwechsel möglich und man muß die Mittelwertanalyse mit den modifizierten kettenbezogenen Parametern ausführen.
 - transformiere die Klassengrößen in Kettengrößen.

Damit sind Netze mit Klassenwechsel und ohne Prioritäten exakt analysierbar.

4.4 Netze mit Klassenwechsel und Prioritäten

Man könnte versucht sein, die vorgestellten Formeln für Prioritätsknoten (siehe Kapitel 3.3.1, 3.3.2) auch auf den Fall mit Klassenwechsel zu übertragen. Dieses Vorgehen ist jedoch nicht möglich, da

- durch den Übergang von Klassen auf Ketten die Dimension des Populationsvektors i.a. kleiner wird und damit ein Ausdruck der Form

$$\underline{k}_{ir} - 1_s$$

keinen Sinn mehr macht, falls man auf eine Komponente $s > U$ zugreift.

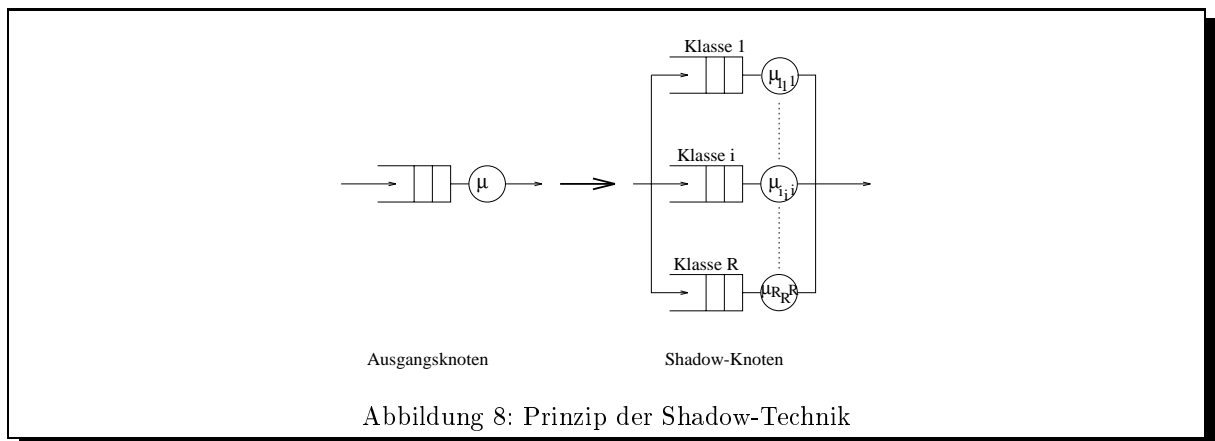
- in einer Kette Aufträge verschiedener Priorität liegen können und somit eine Summation über Klassen höherer Priorität nicht mehr möglich ist.

Man braucht daher ein Verfahren zur Analyse von Prioritätsnetzen, bei dem keine Summation über die Prioritätsklassen erfolgt. Aus diesem Grunde wurde die Idee der Shadow-Technik verwendet [3, 8] und auf Netze mit Klassenwechsel erweitert.

4.4.1 Die Shadow-Technik

Dieses von [16] vorgeschlagene Verfahren für Netze mit einer reinen PR-Strategie geht davon aus, daß man jeden Prioritätsknoten im Netz durch R parallel geschaltete Knoten ersetzt, wobei R die Anzahl der Prioritätsklassen ist. Zur Unterscheidung zwischen Shadow-Knoten und Ausgangsknoten soll für die Shadow-Knoten eine Indizierung der Form (i, r) verwendet werden. Dabei bedeutet

- i : Ausgangsknoten
- j : zum Ausgangsknoten gehöriger Shadow-Knoten
- r : Auftragsklasse



Den Fehler, den man dadurch macht, daß im erweiterten Knoten die Aufträge auch parallel bedient werden können, gleicht man dadurch aus, daß man iterativ die Bedienzeiten für die einzelnen Knoten erhöht. Je niedriger die Priorität eines Auftrags ist, desto höher wird seine Bedienzeit, um so den Effekt der Verdrängung zu simulieren.

Nach Abbruch der Iteration gilt für die Bedienzeiten im Shadow-Knoten

$$s_{i,j} \geq s_{i,j}$$

Da bei der Berechnung der Bedienzeiten keine Summation über die Prioritätsklassen erfolgt, ist dieses Verfahren der Shadow-Server für Prioritätsnetze mit Klassenwechsel geeignet.

4.4.2 Shadow-Algorithmus

1. transformiere das Ausgangsmodell in das Shadow-Modell.
2. setze $\lambda_{i_j,r} = 0$.
3. iteriere
 - (a) berechne die Shadow-Auslastung

$$\tilde{\rho}_{i_j,r} = \begin{cases} \lambda_{i_r,r} \cdot s_{i_r,r} & \text{falls } j = r \\ 0 & \text{sonst} \end{cases}$$

- (b) berechne die Shadow-Bedienzeiten

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}} & \text{falls } j = r \\ 0 & \text{sonst} \end{cases}$$

Hierbei ist $s_{i,r}$ die Originalbedienzeit eines Auftrags der Klasse r am Knoten i im Ausgangsnetz. Für Knoten mit $s_{i_j,r} = 0$ muß die entsprechende Besuchshäufigkeit 0 sein. Die restlichen Besuchshäufigkeiten bleiben unverändert.

- (c) werte das Shadow-Modell mittels der Mittelwertanalyse aus.

D.h. man führt die Mittelwertanalyse mit $s_{i_j,r}$ durch und erhält den für den nächsten Iterationsdurchlauf benötigten Durchsatz $\lambda_{i_j,r}$. Falls sich die $\lambda_{i_j,r}$ in aufeinanderfolgenden Schritten um weniger als ε unterscheiden, brich die Iteration ab. Ansonsten gehe zurück zu Schritt 3.a

4.4.3 Erweiterte Shadow-Technik

Messungen an realen Systemen und Vergleiche mit Simulationsergebnissen haben ergeben, daß die Methode der Shadow-Server keine befriedigenden Resultate liefert. Aus diesem Grunde wurde in [8] eine Erweiterung der Shadow-Methode für Netze ohne Klassenwechsel vorgeschlagen. Diese erweiterte Methode unterscheidet sich vom Ausgangsverfahren dadurch, daß der Ausdruck

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

in der Iteration noch mit einem Korrekturfaktor multipliziert wird. Dadurch wird erreicht, daß die iterativ berechneten Bedienzeiten nicht zu stark ansteigen und man eine bessere Approximation erhält.

Zu diesem Zweck wird der Korrekturfaktor δ wie folgt festgelegt :

$$\delta_{i_j,r} = \begin{cases} \frac{\rho(r)[1 - \rho(r)] + \alpha(r) \cdot \rho(r+1)}{\rho(r)[1 - \rho(r)]^2 + \alpha(r) \cdot \rho_{i_r,r}} & \text{für } r = 2..R \text{ und } j = r \\ 0 & \text{sonst} \end{cases}$$

mit

$$\begin{aligned} \alpha(r) &= \sum_{k=1}^{r-1} \frac{\rho_{i_j,r}}{\omega_{r,k}} \\ \omega_{r,k} &= \frac{s_{i,k}}{s_{i,r}} \\ \rho(r) &= \sum_{k=1}^{r-1} \rho_{i_k,k} \end{aligned}$$

Durch den Korrekturfaktor δ ändert sich nur Schritt 3.b im Shadow-Algorithmus

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}} & \text{falls } j = r \\ 0 & \text{sonst} \end{cases}$$

Sind die Auslastungen der einzelnen Shadow-Stationen sehr nahe bei 1, so kann der Ausdruck

$$1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \cdot \tilde{\rho}_{i_s,s}$$

durch Rechenungenauigkeiten und die approximative Technik einen negativen Wert annehmen. In diesem Fall liefert das Verfahren unsinnige Ergebnisse und muß abgebrochen werden. Ein Ziel der Leistungsbeurteilung ist das Erkennen von Engpaßknoten; das Verfahren erlaubt dann wenigstens zu erkennen, welcher Knoten einen Engpaß bildet.

4.4.4 Erweiterte Shadow-Technik mit Intervallhalbierung

- Problem :

In der ursprünglichen und erweiterten Shadow-Methode kann der Ausdruck

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

sehr nahe bei 1 liegen. Dadurch nimmt die Bedienzeit $s_{i_r,r}$ einen sehr großen Wert an und der Durchsatz durch diesen Knoten fällt sehr stark ab. Damit wird aber

$$\sum_{s=1}^{r-1} \tilde{\rho}_{i_s,s}$$

im nächsten Iterationsschritt sehr klein und das Verfahren beginnt zu schwingen. D.h. die Iteration pendelt immer zwischen einem sehr großen und einem sehr kleinen Wert hin und her. Dieses Problem wurde von den bisherigen Verfahren nicht berücksichtigt. Aus diesem Grunde wurde das Shadow-Verfahren noch um eine neue eigene Variante erweitert, was schließlich zur Shadow-Methode mit Intervallhalbierung führte.

- Lösung

Zur Lösung dieses Problems beachte man, daß der wirkliche Wert der Leistungsgrößen irgendwo zwischen den schwingenden Werten liegen muß. Die Idee ist nun, die Mittelwertanalyse um ein Intervallhalbierungsverfahren zu erweitern. Dieses Verfahren wird aber erst angewendet, wenn erkennbar ist, daß das Ausgangsverfahren schwingt. Zu diesem Zweck wird die Verweilzeit in jedem Schritt als Mittelwert aus der im letzten Iterationsschritt ermittelten schwingenden Verweilzeit und der neu berechneten Verweilzeit gesetzt. Mit dieser gemittelten Verweilzeit wird in der Mittelwertanalyse dann weitergerechnet.

Dadurch wird erreicht, daß man innerhalb weniger Schritte aus dem instabilen Schwingungszustand herauskommt und das Verfahren konvergiert.

4.4.5 Erweiterte Shadow-Technik mit Klassenwechsel

Ist im Netz Klassenwechsel möglich, so werden die Auslastungen und Bedienzeiten der Shadow-Knoten in Klassengrößen berechnet. Daraufhin werden die Netzgrößen in Kettengrößen transformiert und die Netzanalyse auf Kettengrößen durchgeführt. Die erhaltenen Werte transformiert man wieder zurück in Klassengrößen und startet die Shadow-Iteration erneut, bis ein Abbruchkriterium erfüllt ist. Damit erhält man folgenden Algorithmus :

1. bestimme die Ketten im Netz.
2. transformiere das Ausgangsmodell in das Shadow-Modell.
3. setze $\lambda_{i_j,r} = 0$.
4. iteriere
 - (a)

$$\tilde{\rho}_{i_j,r} = \begin{cases} \lambda_{i_r,r} \cdot s_{i_r,r} & \text{falls } j = r \\ 0 & \text{sonst} \end{cases}$$

(b)

$$s_{i_j,r} = \begin{cases} \frac{s_{i,r}}{r-1} & \text{falls } j = r \\ 1 - \sum_{s=1}^{r-1} \frac{1}{\delta_{i_s,s}} \tilde{\rho}_{i_s,s} & \\ 0 & \text{sonst} \end{cases}$$

Hierbei ist $s_{i,r}$ die Originalbedienzeit eines Auftrags der Klasse r am Knoten i im Ausgangsnetz. Für Knoten mit $s_{i_j,r} = 0$ muß die entsprechende Besuchshäufigkeit 0 sein. Die restlichen Besuchshäufigkeiten bleiben unverändert.

- (c) transformiere die Klassengrößen in Kettengrößen (zur Unterscheidung von den Klassengrößen werden diese mit einem * versehen)
- (d) werte das kettentransformierte Shadow-Modell mittels der Mittelwertanalyse aus. Falls die Leistungsgrößen zu schwingen beginnen, wende die Methode der Intervallhalbierung an und setze

$$\bar{t}_{i_j,r} = \frac{\bar{t}_{i_j,r}^{alt} + \bar{t}_{i_j,r}^{neu}}{2}.$$

Mittels dieser neuen Verweilzeit berechne die Leistungsgrößen des Netzes erneut.

- (e) transformiere die Kettengrößen in Klassengrößen.

Falls sich die $\lambda_{i_j,r}^*$ (Kettengröße !) in aufeinanderfolgenden Schritten um weniger als ε unterscheiden, brich die Iteration ab. Ansonsten gehe zurück zu Schritt 4.a

4.4.6 Klassenwechsel und gemischte Prioritätsstrategie mit der Shadow-Technik

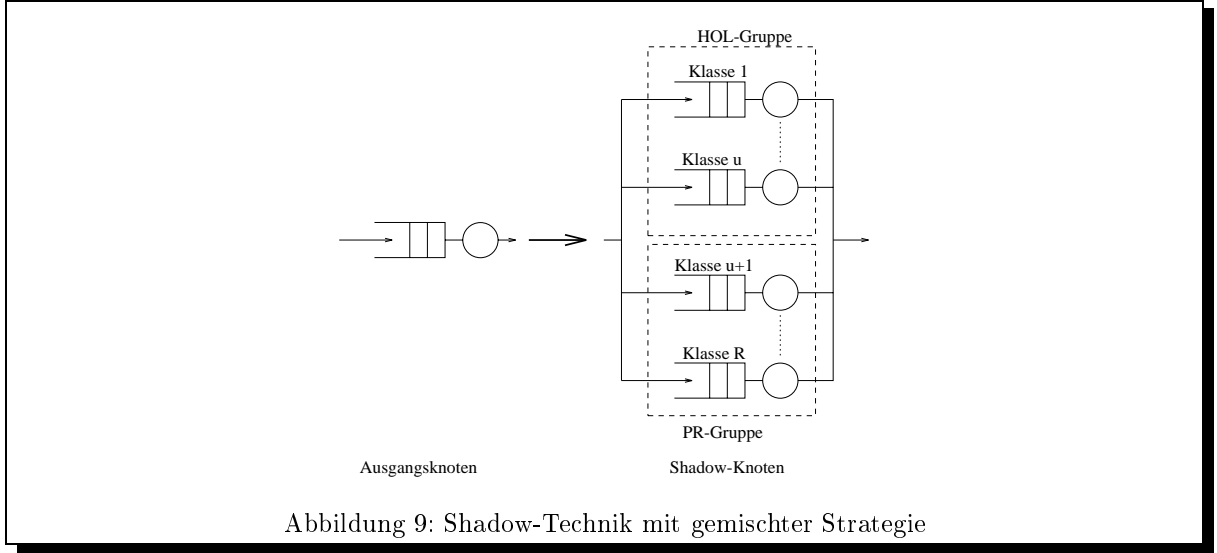
Die von [16] vorgeschlagene Shadow-Technik war nur auf Netze mit der PR-Strategie anwendbar. Bei den in [7] vorgeschlagenen Modellen eines UNIX basierten Multiprozessorbetriebssystems wurde jedoch eine gemischte Prioritätsstrategie verwendet. Zur Berechnung muß die bekannte Shadow-Technik so erweitert werden, daß eine gemischte Strategie möglich ist.

Zu diesem Zweck wurde die bereits vorgeschlagene Idee der Gruppenbildung (siehe Kapitel 3.3.3) angewendet. In Bild 9 ist dieses Konzept dargestellt. Man muß beachten, daß ein ankommender Auftrag der PR-Klasse die Wartezeit eines HOL-Auftrags in der in Bild 9 dargestellten Situation nicht beeinflußt. Dagegen kann ein PR-Auftrag jederzeit durch einen ankommenden HOL-Auftrag verdrängt werden. Dieser Situation wird dadurch Rechnung getragen, daß in der erweiterten Shadow-Methode der Korrekturfaktor δ geeignet modifiziert wird [6].

$$\psi_{i_j,r} = \begin{cases} \frac{\varrho(r)[1 - \varrho(r)] + \beta(r) \cdot \tilde{\varrho}(r)}{\varrho(r)[1 - \varrho(r)]^2 + \beta(r) \cdot \varrho_{i_r,r}} & \text{für } r = 2..R \text{ und } j = r \\ 0 & \text{sonst} \end{cases}$$

$$\beta(r) = \begin{cases} \sum_{k=1, k \neq r}^u \frac{\rho_{i_j r}}{\omega_{rk}} & \text{falls } r \in \text{HOL} \\ \sum_{k=1}^{r-1} \frac{\rho_{i_j r}}{\omega_{rk}} & \text{falls } r \in \text{PR} \end{cases}$$

(Bedeutung der einzelnen Größen siehe Kapitel 4.4.7)



4.4.7 Klassenwechsel und beliebig gemischte Prioritätsstrategie mit der Shadow-Technik

Es wurde bisher angenommen, daß die Klassen $1, \dots, u$ reine HOL-Klassen und die Klassen $u + 1, \dots, R$ reine PR-Klassen sind.

Aus den hergeleiteten Formeln ist erkennbar, daß man bzgl. der benötigten Korrekturfaktoren im HOL-Fall über alle HOL-Klassen ausgenommen der gerade betrachteten Klasse und im PR-Fall über alle Klassen höherer Priorität summiert.

Mit Hilfe dieser Beziehungen ist es nun möglich, die Formeln so zu erweitern, daß auch eine Vermischung von HOL- und PR-Klassen möglich ist. D.h. Klasse 1, 3 und 5 sind z.B. HOL-Klassen, Klasse 2, 4 und 6 dagegen PR-Klassen. Sei dazu ξ_r die Menge aller Auftragsklassen, die nicht durch einen Auftrag der Klasse r verdrängt werden können, wobei zusätzlich gilt, daß $r \notin \xi_r$ ist. Für die Berechnung der Korrekturfaktoren hat diese Prioritätenvermischung die Konsequenz, daß

- für eine HOL-Klasse r die Summationen nicht von 1 bis u , sondern über alle Prioritätsklassen $s \in \xi_r$ laufen.
- für eine PR-Klasse r die Summationen nicht von 1 bis $r - 1$, sondern über alle Prioritätsklassen $s \in \xi_r$ laufen.

Für eine beliebig gemischte Prioritätsstrategie erhält man damit

$$s_{i_j, r} = \begin{cases} \frac{s_{i, r}}{1 - \sum_{s \in \xi_r} \frac{1}{\psi_{i_s, s}} \cdot \tilde{\rho}_{i_s, s}} & \text{falls } r = j \\ 0 & \text{sonst} \end{cases}$$

$$\psi_{i_j, r} = \begin{cases} \frac{\varrho(r)[1 - \varrho(r)] + \beta(r) \cdot \tilde{\varrho}(r)}{\varrho(r)[1 - \varrho(r)]^2 + \beta(r) \cdot \varrho_{i_r, r}} & \text{falls } r = j \\ 0 & \text{sonst} \end{cases}$$

$$\beta(r) = \sum_{k \in \xi_r} \frac{\rho_{i_j, r}}{\omega_{r, k}}$$

$$\omega_{r, k} = \frac{s_{ik}}{s_{ir}}$$

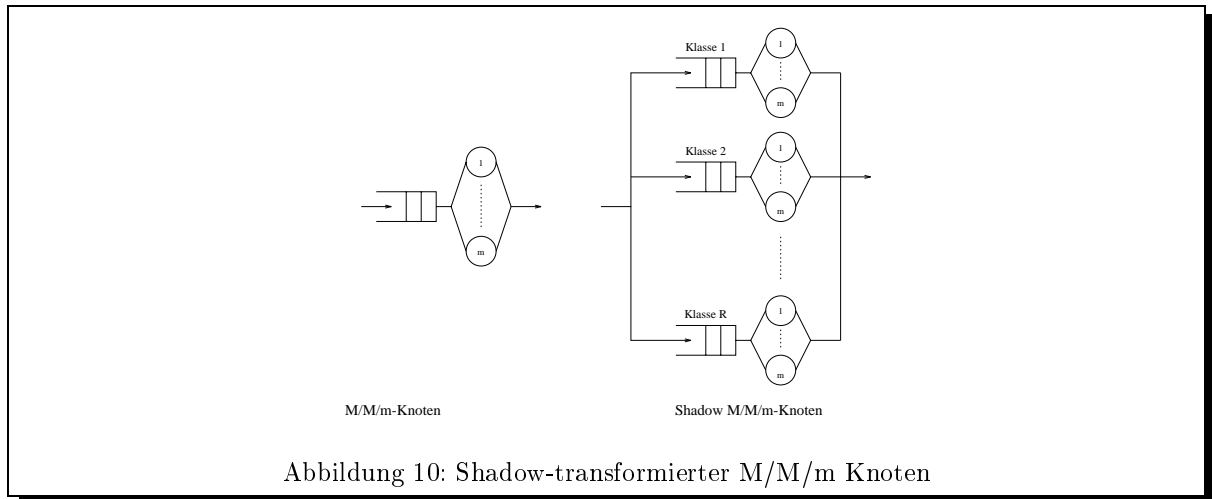
$$\varrho(r) = \sum_{k \in \xi_r} \rho_{i_k, k}$$

$$\tilde{\varrho}(r) = \sum_{k \in \{\xi_r \cup r\}} \varrho_{i_k, k}$$

4.4.8 Shadow-Technik und erweiterter M/M/m-Knoten

In den nun folgenden Betrachtungen soll ein neuer Knotentyp - der sog. erweiterte M/M/m-Knoten - eingeführt werden. Dieser Knotentyp spielt vor allem bei den zu betrachtenden UNIX-Betriebssystemmodellen eine wichtige Rolle.

Im Falle eines M/M/m-Knotens ist die Analyse sehr einfach. Man muß nur den M/M/m-Knoten in seine Shadow-Form transformieren (siehe Bild 10) und die Formeln für M/M/m-Knoten verwenden, wenn man den Knoten mittels der Mittelwertanalyse in jeder Shadow-Iteration analysiert.

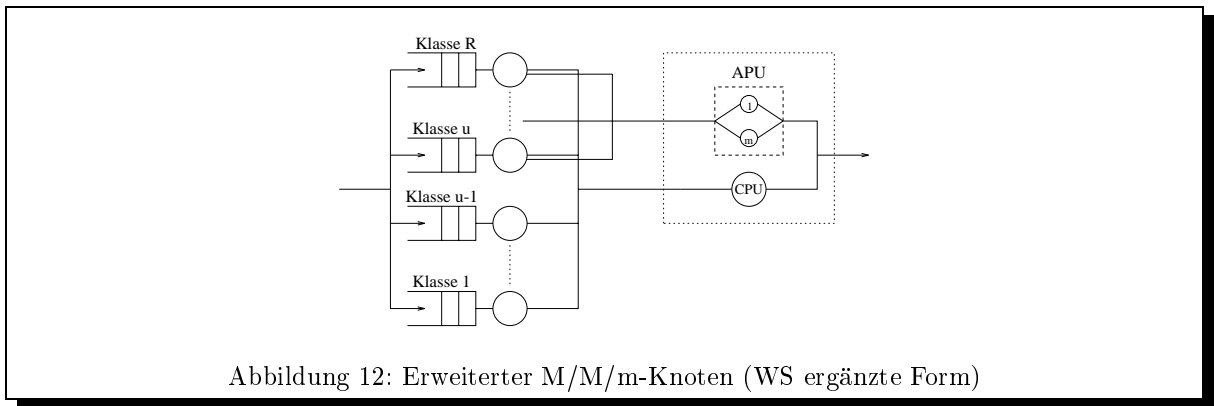
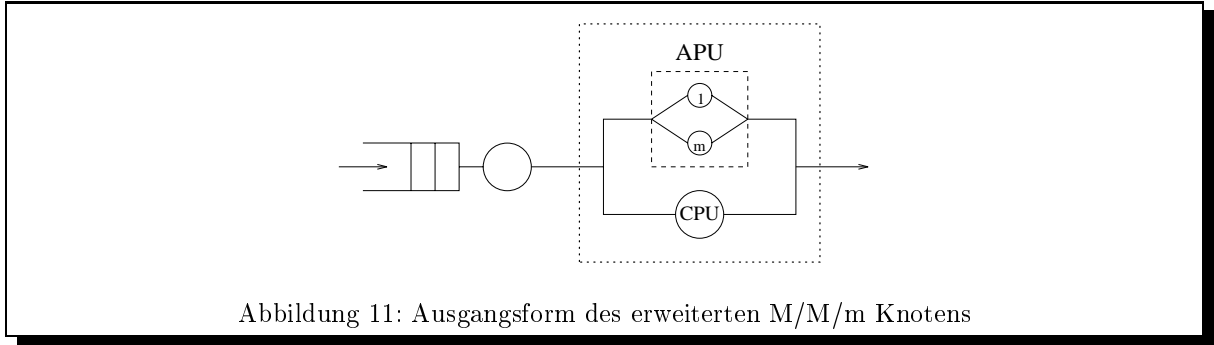


Der Unterschied zwischen dem Knoten in Bild 10 und dem erweiterten M/M/m-Knoten in Bild 11 ist, daß man im erweiterten M/M/m-Knoten R Jobklassen hat, die nach Prioritäten geordnet sind. In der CPU können alle Auftragsklassen bearbeitet werden, während in der sog. APU (Associated Processing Unit) nur die Auftragsklassen $u..R$ ($1 \leq u \leq R$) bearbeitet werden können. Im M/M/m-Knoten kann dagegen jeder Job aus jeder Jobklasse jeden freien Server belegen.

Aus Gründen der besseren Übersichtlichkeit soll für jede Auftragsklasse eine eigene Warteschlange eingeführt werden. Man gelangt dann zur Darstellung aus Bild 12

Ist z.B. $u = R = 3$, so können in der CPU Aufträge der Klasse 1, 2, 3 bearbeitet werden, während in der APU nur die Auftragsklasse 3 bearbeitet werden kann. In der CPU können Aufträge der Klasse 3 jederzeit verdrängt werden, auf der APU jedoch nicht, da auf der APU nur Klasse 3 Aufträge bearbeitet werden.

Für $1 < u \leq R$ wird auf der APU also die Prioritätsreihenfolge verändert, Klasse u hat höchste Priorität, ..., Klasse R hat niedrigste Priorität. Diese Situation kann in einem gewöhnlichen M/M/m-Knoten nicht mehr erfaßt werden. Man muß aber auch berücksichtigen, daß auf den APU's ebenfalls eine Verdrängung stattfindet, wenn mehr als eine Auftragsklasse die APU betreten darf. Um diesen Effekt der Verdrängung in der APU zu berücksichtigen, wendet man auf jeden Knoten, von dem aus ein Übergang in die APU



möglich ist, nochmals die Idee der Shadow-Server an und erhält schließlich die in Bild 13 gezeigte endgültige Darstellung des erweiterten M/M/m-Knotens.

Die Komponenten APU_u, \dots, APU_R bilden die ursprüngliche Ausgangs-APU. Auf das so modifizierte Netz kann man dann die erweiterte Shadow-Technik für gemischte Prioritätsstrategien anwenden. D.h. die Bedienzeiten an den einzelnen Knoten werden erhöht, um den Effekt der Verdrängung zu simulieren. Da aber in der APU eine andere Prioritätsreihenfolge vorliegt als in der CPU (in der APU ist die Priorität eines Auftrags i.a. höher als in der CPU), muß die Bedienzeit für APU_i in der vollständig shadowisierten Form ebenfalls erhöht werden. Es muß daher die Shadow-Iteration auch getrennt für APU's und CPU's durchgeführt werden.

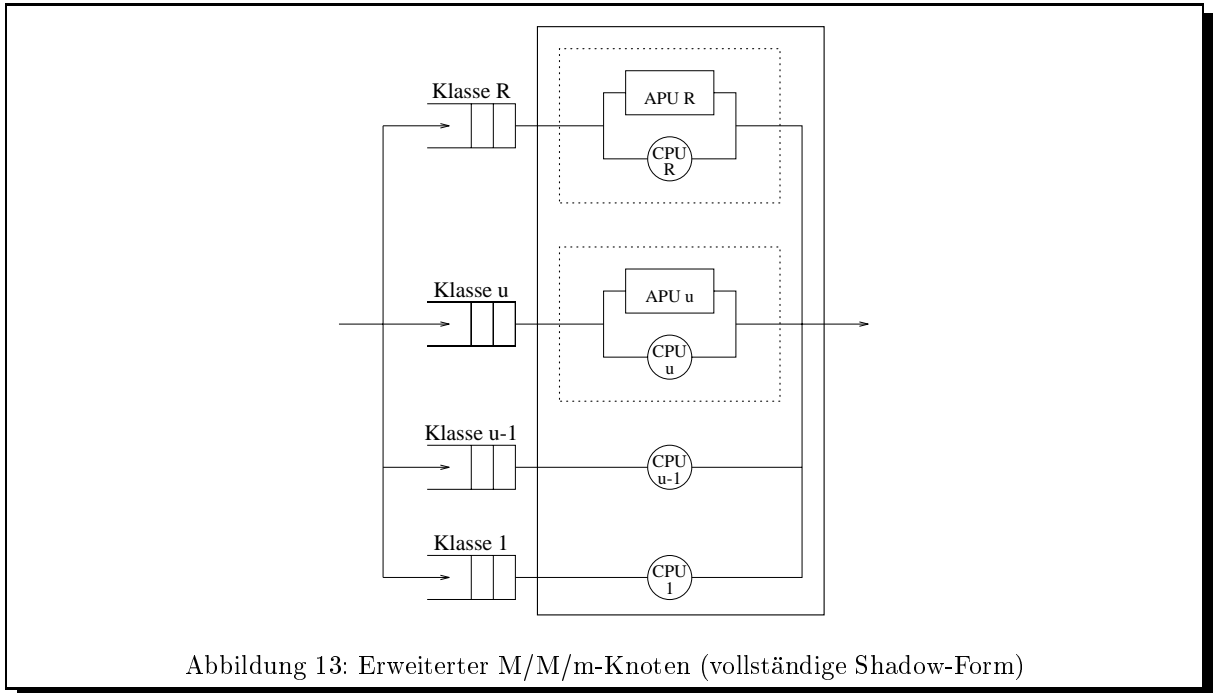
Sei ζ die Menge, welche die höchste Priorität eines Auftrags enthält, der die Bedienstation (CPU, APU) betreten kann. Im allgemeinen Fall eines verallgemeinerten erweiterten M/M/m-Knotens [6] kann ζ auch mehr als zwei Elemente enthalten.. In [5, 6] wird gezeigt, daß im Shadow-Algorithmus nur der Schritt zur Berechnung der mittleren Bedienzeit $s_{i,j,r}$ geändert werden muß:

$$\forall c \in \zeta \quad : \quad s_{i,j,r} = \begin{cases} \frac{s_{ir}}{1 - \sum_{s \geq c, s \in \xi_r} \frac{1}{\psi_{i_s, s}} \cdot \tilde{\rho}_{i_s, s}} & \text{falls } r = j \\ 0 & \text{sonst} \end{cases}$$

Da die Iterationen getrennt über CPU und APU durchgeführt werden, erhält man i.a. einen sog. asymmetrische Knoten, d.h. die Bedienzeiten der einzelnen Bedieneinheiten des Knotens sind verschieden.

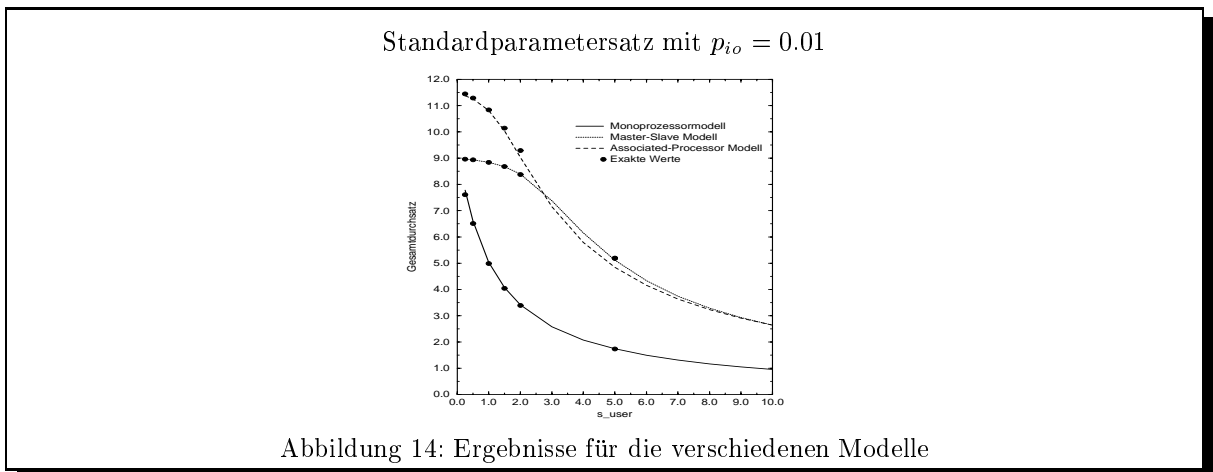
5 Bewertung und Validierung der Verfahren

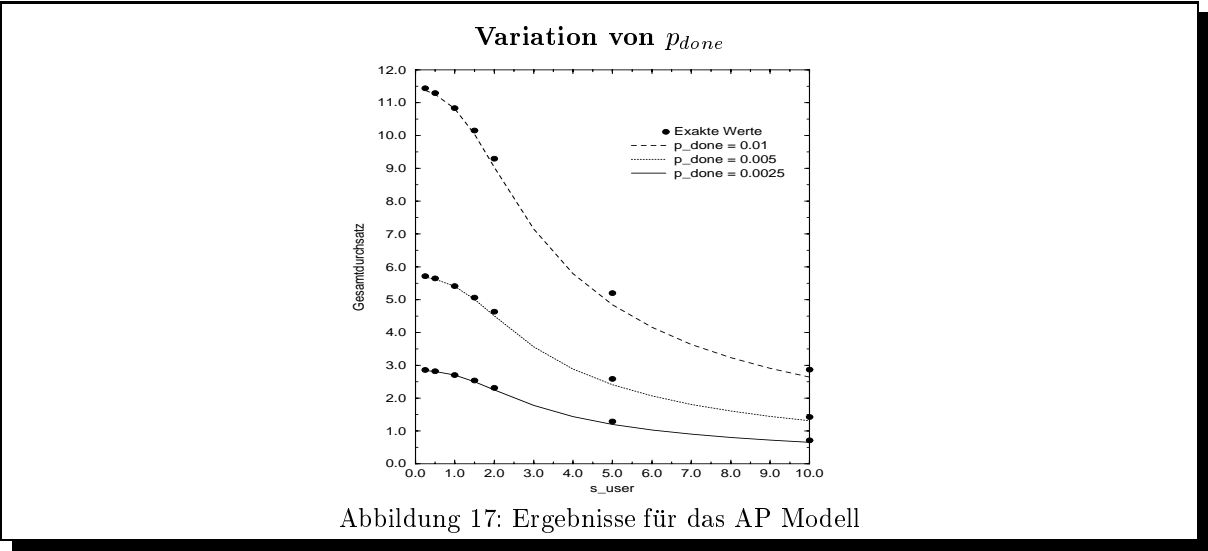
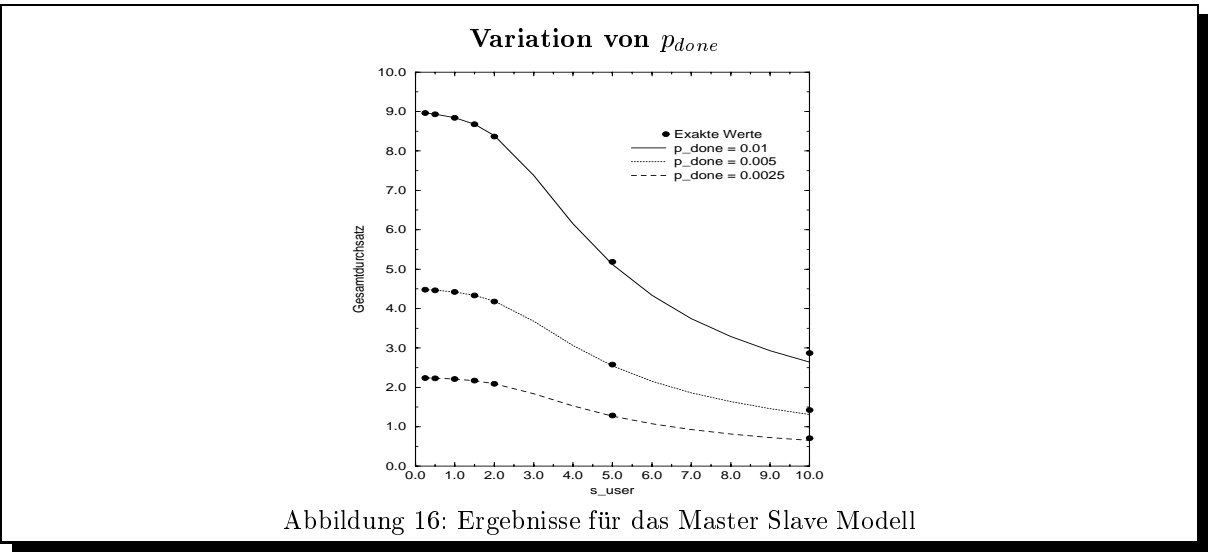
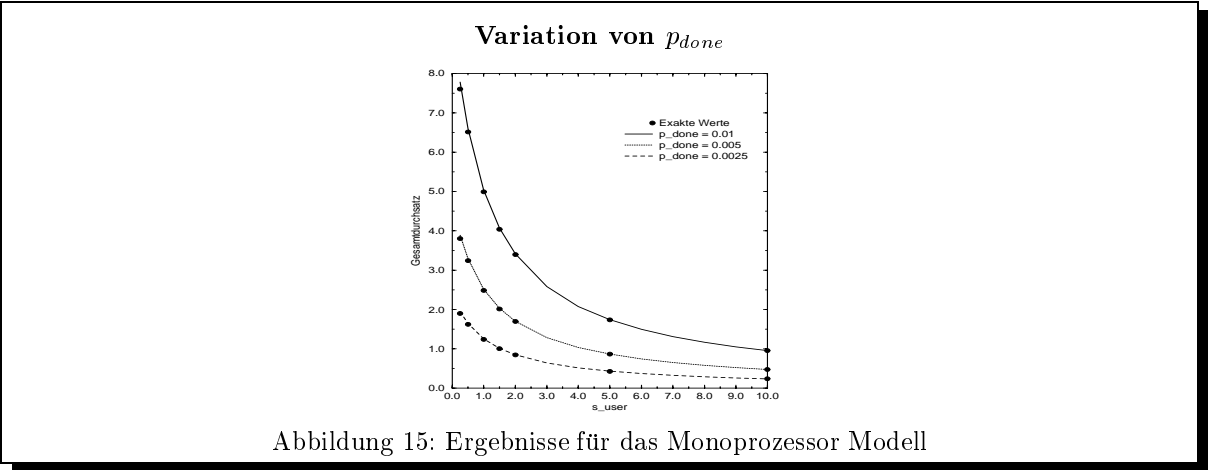
Die Anwendung der vorgestellten Verfahren soll hier exemplarisch an den Jung'schen Modellen [7] eines UNIX basierten Multiprozessorbetriebssystems (siehe Kapitel 1.3) aufgezeigt werden.



Bei diesen Modellen wurden die Parametersätze in weiten Bereichen variiert, wobei aufgrund von Messungen und numerischen Analysen bereits validierte Vergleichsergebnisse zur Verfügung standen.

5.1 Graphische Darstellung der Ergebnisse





Wie aus den Ergebnissen ersichtlich ist, sind für den gewählten Parametersatz die Unterschiede zwischen den exakten Werten und den analytischen Werten sehr gering. Der Systemdurchsatz ist wie erwartet beim

Associated Processor Modell am höchsten. Bei genaueren Untersuchungen wurde festgestellt, daß beim Master Slave Modell der Systemdurchsatz durch hinzufügen weiterer APU's kaum noch weiter gesteigert werden kann, weil die CPU einen Engpaß darstellt, wenn sich die Aufträge häufig im Kernkontext oder Treiberkontext befinden, die nicht auf den APU's bearbeitet werden können. Für solche Systeme ist dann das Associated-Processor System - oder wenn es die Hardware erlaubt, ein vollkommen paralleles System, bei dem auch die Aufträge im Treiberkontext parallel bearbeitet werden können - sinnvoll.

Natürlich muß für I/O-intensive Aufgaben die Anzahl peripheren Geräte mit der Anzahl der Prozessoren zunehmen, da sonst die Prozessoren auf die Fertigstellung von I/O-Aufträgen warten müssen und somit der Leistungsgewinn durch den Einsatz des Multiprozessors wieder zunichte gemacht wird. Ein weiteres sehr wichtiges Ergebnis der Untersuchungen war, daß sich für $s_{user} \geq 2$ das Master Slave Modell und das AP Modell fast identisch verhalten und nur für $s_{user} < 2$ ist das AP Modell merklich besser. D.h. bei längeren Useraufträgen lohnt sich die aufwendige Parallelisierung des Kerns nicht. Da UNIX aber sehr betriebsystemintensiv ist, der Rechner bearbeitet ungefähr die Hälfte der Zeit Betriebssystemaufträge ($s_{kern} \approx s_{user}$), lohnt sich die Parallelisierung des Kerns meistens doch [7]. Weiterhin wurden auch Untersuchungen bzgl. der Anzahl der APU's und der Leistungsfähigkeit der IO durchgeführt. Dabei hat sich ergeben, daß bei einem Master-Slave Ansatz spätestens mit zwei APU's die obere Leistungsgrenze erreicht ist.

5.2 Einfluß der Prioritäten

In der folgenden Tabelle wird der Einfluss der Prioritäten für das Master Slave Modell gezeigt. Hierbei ist s_{user} die Bedienzeit für User-Aufträge. p_{io} wird variiert, während p_{done} konstant gehalten wird. Auf der rechten Seite der Tabelle werden die Werte für das Netzwerk ohne Prioritäten (nonpre), die Originalwerte (orig) und die approximativen Werte (approx), welche wir mit unserer neuen Methode erhalten haben, aufgezeigt.

s_{user}	p_{io}	p_{done}	Throughput		
			nonpre	approx	orig
1.00	0.9	0.01	0.6593	0.6594	0.6593
1.00	0.01	0.01	8.2745	9.8736	9.8716
1.50	0.9	0.01	0.6593	0.6594	0.6592
1.50	0.01	0.01	7.6517	9.8194	9.7666
2.00	0.9	0.01	0.6593	0.6594	0.6592
2.00	0.01	0.01	7.1078	9.5178	9.4700
2.50	0.9	0.01	0.6592	0.6593	0.6592
2.50	0.01	0.01	6.6170	8.8123	8.6144

Tabelle 6: Einfluß der Prioritäten

Wie man aus den Ergebnissen ersehen kann, ist der Einfluss der Prioritäten für diese Parameterwerte nicht vernachlässigbar und beeinflusst die Systemleistung beträchtlich.

6 Folgerung und Ausblick

Mit Hilfe der neuen Technik ist es möglich, die vorgestellten Betriebssystemmodelle in Sekundenschnelle zu analysieren. Dabei waren die Unterschiede zwischen den Originalergebnissen (welche wir durch Messungen am realen System erhalten haben) und den berechneten Ergebnissen sehr gering [6]. Ein weiterer Vorteil der neuen Technik ist, daß man nicht nur auf die Mittelwertanalyse beschränkt ist, sondern auch andere Techniken wie SCAT, der weit weniger Speicherplatz und Rechenzeit benötigt wie die Mittelwertanalyse und trotzdem ausreichend genau ist, oder Faltung verwenden kann.

Um die vorgestellten UNIX-Modelle noch realistischer zu machen, kann man ein offenes Warteschlangennetz annehmen. Um diese Art von Netzen zu lösen, kann man beispielsweise die Schließmethode [4] verwenden. Hierbei wird das offene Netz in ein geschlossenes Netz transformiert, welches dann mittels unserer neuen Technik analysiert werden kann.

Bei Netzen mit allgemein verteilten Bedienzeiten, kann man als Standard-Analyseverfahren in unserer neuen Technik beispielsweise die Methode von Marie [12] verwenden. Dabei wird das Netz ebenfalls in seine

Shadow-Form transformiert und analysiert. Zwischen zwei Shadow-Iterationen wird die Methode von Marie verwendet, um die benötigten Leistungsgrößen zu berechnen, die man für den nächsten Iterationsschritt benötigt. Noch realitätsnäher sind offene Warteschlangenmodelle mit allgemein verteilten Bedienzeiten. Eine Möglichkeit zur Analyse dieser Modelle ist, eine Erweiterung der Methode von Kühn [10] als Standard-Analyseverfahren in unserer neuen Technik zu verwenden. Die andere Möglichkeit ist, das Netz zunächst mittels der Schließmethode in ein geschlossenes Netz zu transformieren und dieses dann mittels der neuen Technik unter Verwendung der Methode von Marie als Standard-Analyseverfahren zu analysieren.

Aufgrund der sehr guten Resultate dieser neuen approximativen Technik ist geplant, auch für andere Betriebssysteme (z.B. MACH, Lotus, Realzeitbetriebssysteme) entsprechend erweiterte Warteschlangenmodelle zu entwickeln und zu analysieren. Zur Zeit modellieren wir das Betriebssystem BS 2000, um es mit den entwickelten Techniken zu analysieren. Wenn man die Modelle von BS2000 und UNIX miteinander vergleicht, stellt man fest, daß diese sehr ähnlich sind. Es scheint also möglich zu sein, daß man für eine größere Zahl von Betriebssystemen ein gemeinsames Grundmodell angeben kann.

Literatur

- [1] Bryant R.M.; Krzesinski A.E.; Lakshmi M.S.; Chandy K.M.: The MVA Priority Approximation, ACM Transactions on Computer Systems, Vol. 2, No. 4, pp. 335 - 359, Nov. 1984.
- [2] Bolch G.: Leistungsbewertung von Rechensystemen, Teubner, 1989.
- [3] Bruell S.C., Balbo G.: Computational Algorithms for Closed Queueing Networks, North Holland, 1980.
- [4] Bolch G., Gaebell M., Jung. H.: Analyse offener Warteschlangennetze mit Methoden für geschlossene Warteschlangennetze, Proceedings der DGOR-Jahrestagung, Aachen, Sept. 1992.
- [5] Bolch G., Greiner S.: Approximative analytische Leistungsbewertung am Beispiel eines UNIX-basierten Multiprozessor-Betriebssystems, Interner Bericht TR-I4-1-94, Universität Erlangen-Nürnberg, IMMD4.
- [6] Greiner S.: Leistungsbewertung des Betriebssystems UNIX mit Hilfe approximativer analytischer Methoden, Studienarbeit am IMMD IV, Universität Erlangen, 1993.
- [7] Jung H.: Leistungsbewertung Unix basierter Betriebssysteme für Multiprozessoren mit globalem Speicher, Dissertation am IMMD IV, Universität Erlangen, 1991.
- [8] Kaufmann J.S.: Approximation Methods for Networks of Queues with Priorities, Performance Evaluation Vol 4, pp. 183 - 198, 1984.
- [9] Kirschnick M.: The Performance Evaluation and Prediction System for Queueing Networks PEPSY-QNS, Technical Report TR-I4-18-94 of the Computer Science Department, IMMD IV, Operating Systems, University Erlangen, Germany, 1994
- [10] Kuehn P.J.: Approximate Analysis of General Queueing Networks by Decomposition, IEEE Transactions on Communications, Vol. 27, No1, pp. 113-126, Jan. 1979.
- [11] Lazowska E.D. , Zahorjan J. , Graham G.S. , Sevcik K.C.: Quantitative System Performance - Computer System Analysis Using Queueing Network Models, Prentice Hall, 1984.
- [12] Marie R.: An Approximate Analytical Method for General Queueing Networks IEEE Transactions on Software Engineering, Vol.5, No.5, pp.530-538, Sept. 1979.
- [13] Muntz R.R.: Poisson Departure Process and Queueing Networks, Proc. of the 7th Annual Princeton Conf. on Information Sciences and Systems, Princeton University, pp. 435 - 440 March 1973.
- [14] Reiser M. , Lavenberg S.S.: Mean-Value-Analysis of Closed Multichain Queueing Networks, Journal of the ACM, Vol 27 , No. 2 , pp 313 - 322 . April 1980.
- [15] Sauer C.H., Chandy K.M.: Computer System Performance Modelling, Prentice Hall, 1981.
- [16] Sevcik K.C.: Priority Scheduling Disciplines in Queueing Network Models of Computer Systems, Proc. IFIP Congress, North Holland, pp 565 - 570 , 1977.