ZUR OPTIMALEN STEUERUNG DES MULTIPROGRAMMINGGRADES IN RECHNERSYSTEMEN MIT VIRTUELLEM SPEICHER UND PAGING

Paul Kühn

Institut für Nachrichtenvermittlung und Datenverarbeitung
Universität Stuttgart

1. PROBLEMSTELLUNG

Heutige Großrechnersysteme für den technisch-wissenschaftlichen wie auch für den kommerziellen Einsatz sind durch den Aufbau einer Speicherhierarchie gekennzeichnet. Hierbei werden in der untersten Ebene schnelle, in ihrer Kapazität jedoch begrenzte Speichermedien eingesetzt, welche an die Verarbeitungsgeschwindigkeit schneller Prozessoren angepaßt sind. Massendaten werden dagegen in darüberliegenden Ebenen aus Speichern größerer Kapazität, aber geringerer Zugriffsgeschwindigkeit gehalten. Diese Konfiguration verbindet kleine Zugriffszeiten mit großer Speicherkapazität in wirtschaftlicher Weise, hat aber andererseits einen umfangreichen Datenaustausch zwischen verschiedenen Ebenen der Speicherhierarchie zur Folge, welcher u.U. zu Engpässen führen kann.

Durch die Virtualisierung des Speichers wird jedem Benutzerprogramm ein zusammenhängender Adressraum zur Verfügung gestellt, welcher i.a.wesentlich größer ist als der reale Adressraum des Arbeitsspeichers. Zum Ablauf der einzelnen Programme müssen daher nacheinander Teile der Programme in den Arbeitsspeicher geladen werden. Dies erfolgt i.a. nach dem sog. Paging-Verfahren, bei welchem der Datenaustausch nur in Blöcken gleicher Größe erfolgt; hierzu wird der virtuelle Adressraum in Seiten (pages) und dementsprechend auch der Arbeitsspeicher in Rahmen (page frames) unterteilt, wobei ein Seitenrahmen genau eine Seite aufnehmen kann. Um einen sinnvollen Ablauf der Programme zu gewährleisten, sollte stets eine arbeitsfähige Menge von Seiten eines Programmes im Arbeitsspeicher verfügbar sein. Bei Auftreten einer Seitenreferenz bezüglich einer momentan nicht im Arbeitsspeicher befindlichen Seite (page fault) erfolgt eine Unterbrechung, und der Ablauf des unterbrochenen Programmes kann frühestens erst dann fortgesetzt werden, wenn die fällige Seitennachladung erfolgt ist.

Um die Verarbeitungsgeschwindigkeit schneller Prozessoren besser auszunutzen, wird das Prinzip der Parallelarbeit von Prozessoren und Kanälen angewendet. Es werden hierbei mehrere Programme in den Arbeitsspeicher

Erschienen in: Lecture Notes in Computer Science, No. 34
Springer-Verlag, Berlin/Heidelberg/New York (1975),
S. 567 - 580

geladen. Infolge beschränkter Speicherkapazität ist es i.a. erforderlich,diese Programme nur teilweise zu laden. Auf diese Weise kann sich der Prozessor nach dem Auftreten einer Programmunterbrechung infolge eines page faults der Ausführung eines anderen (ablaufbereiten) Programmes widmen, während parallel dazu die Seitennachladung des soeben unterbrochenen Programmes ausgeführt wird (Multiprogramming).

Das Ablaufgeschehen innerhalb von Rechnersystemen mit virtuellem Speicherprinzip hängt u.a. von folgenden Einflußgrößen ab [1,2,3]:

1. Parameter der Rechnerstruktur
   - Arbeitsspeicher-Größe
   - Verarbeitungsgeschwindigkeit von Prozessoren
   - Zugriffszeiten zu Speichern
2. Betriebssystem-Strategien
   - Multiprogramminggrad
   - Speicherplatzverwaltung
   - Seitenholstrategie
   - Seitenersetzungsstrategie
3. Programmeigenschaften
   - Größe der Programme (Seitenzahl)
   - Ausführungsdauer der Programme (Prozessorzeit)
   - Lokalitätseigenschaften der Seitenreferenzen.

Um einen möglichst großen Durchsatz des Systems zu erzielen, müssen die Betriebsmittel des Rechnersystems ausgewogen dimensioniert und verwaltet werden. Insbesondere ist durch die Betriebsmittel-Verwaltung der Effekt des "Seitenflatterns" (thrashing) zu vermeiden, bei welchem der Prozessor infolge uneffektiv laufender Programme entweder unterbeschäftigt ist oder, durch häufige Unterbrechungen bedingt, nützliche Rechenkapazität für Systemprogrammzeiten verbraucht (system overhead).

Ziel der Untersuchungen ist es, Bedingungen für eine Durchsatz-optimale Steuerung des Multiprogramminggrades anzugeben, wobei die wesentlichsten Einflußgrößen berücksichtigt werden. Dies erfolgt anhand einer exakten Analyse eines geschlossenen Warteschlangenmodells, in welches neben realistischen "Bedienungszeiten" wesentliche Merkmale realer Programme und Betriebssysteme einbezogen werden. Als Ergebnisse werden gewonnen:

- Auslastungen des Prozessors durch Benutzer- bzw. Systemprogramme
- Durchsatz
- Auslastungen von Kanälen
- Mittlere Warteschlangenlängen und mittlere Wartezeiten
- Mittlere Durchlaufzeiten der Programme.

Insbesondere wird aus den Ergebnissen dieser Untersuchung der Zusammenhang zwischen Systemauslastung und Multiprogramminggrad für Programmtypen mit sehr unterschiedlichem Lokalitätsverhalten deutlich.

2. VORAUSSETZUNGEN UND MODELLIERUNG

2.1 Rechnerstruktur

Die Grundkonfiguration des Rechnersystems umfaßt einen Rechnerkern CPU (central processing unit) und eine zweistufige Speicherhierarchie bestehend aus einem Arbeitsspeicher ASP und einem Hintergrundspeicher HSP, vergl. Bild 1.

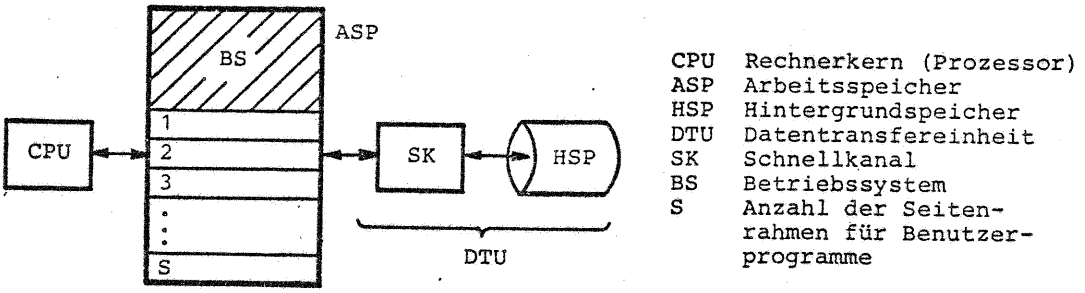

Bild 1. Systemkonfiguration eines Rechnersystems mit zweistufiger Speicherhierarchie

Der Arbeitsspeicher enthält neben den residenten Systemprogrammteilen des Betriebssystems S gleichgroße Seitenrahmen für Benutzerprogramme. Der Hintergrundspeicher HSP (Trommel, Platte) ist über einen Schnellkanal SK mit dem Arbeitsspeicher verbunden. Schnellkanal und Hintergrundspeicher werden im folgenden zur DTU (data transfer unit) zusammengefaßt. Der Ein-/Ausgabeverkehr erfolge über einen E/A-Prozessor und wird im folgenden nicht weiter betrachtet. Es wird davon ausgegangen, daß alle Benutzerprogramme vor und nach ihrer Bearbeitung auf dem Hintergrundspeicher stehen.

2.2 Betriebssystem-Strategien

a) Multiprogramminggrad

Es wird bei den Untersuchungen von einem beliebigen, doch jeweils konstanten Multiprogramminggrad M ausgegangen.

b) Speicherplatzverwaltung (memory management policy)

Die Platzaufteilung des Arbeitsspeichers unter die am Multiprogramming beteiligten Programme sei fest (fixed partitioning), d.h. jedes aktivierte Programm erhält einen festen Teil des Arbeitsspeichers für sich. Darüberhinaus wird vereinfachend angenommen, daß alle Programme denselben ASP-Anteil S/M erhalten (balanced partitioning).

c) Seitenholstrategie (page fetch strategy)

Als Seitenholstrategie wird demand paging vorausgesetzt, d.h. es wird eine Seitennachladung nur nach dem Auftreten eines page faults initiiert.

d) Seitenersetzungsstrategie (page replacement strategy)
Die Seitenersetzungsstrategie bestimmt diejenige Seite, welche bei Seitennachladung aus dem ASP verdrängt wird, falls kein freier Seitenrahmen verfügbar ist. Es werden nur lokal wirkende Strategien zugrundegelegt, welche sich nur auf Seiten des betreffenden Programmes beziehen. Ferner wird vorausgesetzt, daß für die Untersuchungen nur Ersetzungsstrategien für feste Speicherplatzaufteilung zugelassen werden sollen wie z.B. LRU (least recently used: es wird diejenige Seite verdrängt, deren letzter Zugriff am weitesten zurück liegt), FIFO (first-in, first-out: es wird diejenige Seite verdrängt, welche zuerst in den ASP geladen wurde) oder RANDOM (es wird eine zufällig bestimmte Seite verdrängt). Für variable Speicherplatzaufteilung (variable partitioning), wie etwa bei dem Working Set-Ersetzungsalgorithmus, gelten die Überlegungen nur näherungsweise. Die Näherung stimmt jedoch umso besser, je weniger die Speicherplatzaufteilung schwankt.

2.3 Programmeigenschaften

a) Größe der Programme
Benutzerprogramme haben i.a. eine beliebige Anzahl L von Seiten. Vereinfachend wird jedoch angenommen, daß alle am Multiprogramming beteiligten Programme gleich groß sind (konstante Programmgröße).

b) Ausführungsdauern
Die Gesamt-Ausführungsdauern der Programme durch die CPU, d.h. alle CPU-Rechenphasen der Programme jeweils zusammengenommen, seien hyperexponentiell verteilt. Die Verteilungsfunktion (VF) ist festlegbar durch den Mittelwert der CPU-Gesamt-Ausführungsdauer h_{GR} und den Variationskoeffizienten c_{GR}. Diese Voraussetzung bedeutet allerdings keine wesentliche Einschränkung, da die VF der CPU-Gesamt-Ausführungsdauern wegen des konstant angenommenen Multiprogramminggrades und der in 2.3.c angenommenen gleichen Lokalitätseigenschaften der Programme praktisch keinen Einfluß auf den Durchsatz hat.

Im Gegensatz dazu hat jedoch die VF der einzelnen Phasen für die CPU-Rechenzeit zwischen zwei page faults einen entscheidenden Einfluß. Sie wird, in Übereinstimmung mit Messungen an realen Systemen, ebenfalls hyperexponentiell angenommen mit Mittelwert h_{1R} und Variationskoeffizient c_{1R}. Die mittlere Ausführungsdauer der CPU bezüglich einer Seitenreferenz sei h_{1Ref}; sie setzt sich im wesentlichen aus Zugriffszeiten zum Arbeitsspeicher und Befehlsausführungsdauern zusammen.

Die mittlere Gesamt-Ausführungsdauer eines Programmes durch das System (mittlere Durchlaufzeit) t_F setzt sich aus den einzelnen CPU- und DTU-Phasen bzw. -Wartezeiten zusammen; sie ist eine Ergebnisgröße.

c) Lokalitätseigenschaften der Seitenreferenzen

Programme weisen eine mehr oder weniger stark ausgeprägte "Lokalität" bezüglich der zeitlichen Häufung ihrer Zugriffe zu einzelnen Seiten auf. Dieses Lokalitätsverhalten kann meßtechnisch ermittelt werden in Form spezieller Charakteristiken wie

- Working Set-Charakteristik [4]

  Relative Häufigkeit der Zugriffe auf Seiten außerhalb des momentanen Working Set in Abhängigkeit einer "Fensterbreite" τ. Der Working Set ist die Menge derjenigen Seiten, auf welche während der letzten τ Seitenreferenzen zugegriffen wurde.

- Fehlseiten-Charakteristik (page fault rate function) [1]

  Relative Häufigkeit f von page faults in Abhängigkeit des relativen Anteils x von ASP-Seiten eines Programmes bezogen auf dessen Gesamt-Programmgröße L sowie der Seitenersetzungsstrategie.

Für die Untersuchungen werden gemessene Fehlseiten-Charakteristiken zugrundegelegt, welche für verschiedene Programmtypen relativ gut bekannt sind. Bild 2 zeigt sechs Beispiele für verschiedene technisch-wissenschaftliche und kommerzielle Programmtypen [5,6]. Zur Untersuchung wird vereinfachend angenommen, daß alle Programme jeweils dieselbe Fehlseiten-Charakteristik besitzen.

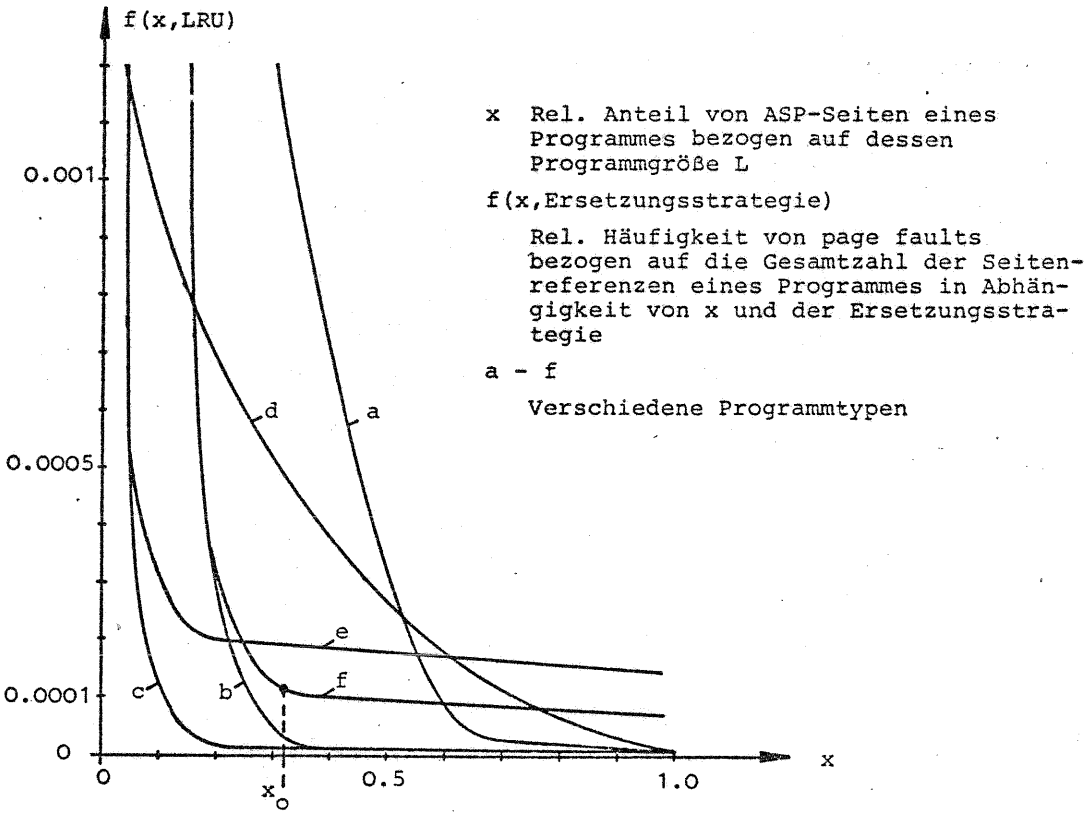

Bild 2. Fehlseiten-Charakteristiken verschiedener Programmtypen

2.4 Warteschlangenmodell

Die analytische Beschreibung des Ablaufgeschehens erfolgt mit Hilfe eines geschlossenen Warteschlangenmodells [3], vergl. Bild 3.

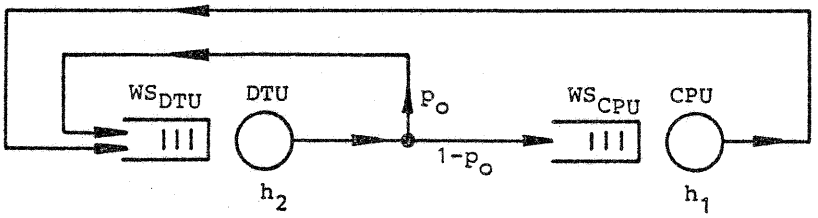

Bild 3. Geschlossenes Warteschlangenmodell

Das Warteschlangenmodell nach Bild 3 besitzt zwei Bedienungseinheiten CPU und DTU, welche die Ausführungsdauern eines laufenden Programmes bis zum nächsten page fault bzw. die Belegungsdauern je Seitennachladung charakterisieren. Die Warteschlangen WS_{CPU} bzw. WS_{DTU} enthalten Anforderungen ablaufbereiter Programme an die CPU bzw. Anforderungen unterbrochener Programme an die DTU zwecks Seitennachladung. Die Verzweigung nach der DTU mit Wahrscheinlichkeit p_o charakterisiert das Bearbeitungsende eines Programmes. In dem geschlossenen Modell, in welchem eine konstante Anzahl M von Anforderungen zirkuliert entsprechend einem konstanten Multiprogramminggrad M, charakterisiert die mit p_o abgezweigte Rückkopplung nach der DTU, daß nach dem Bearbeitungsende eines Programmes momentan ein neues Programm aktiviert wird.

Die Bearbeitungsfolge eines Programmes kann mit den in Bild 4 dargestellten Programmzuständen beschrieben werden. In Bild 4 ist bereits der Zustand berücksichtigt worden, in welchem nach Auftreten einer Unterbrechung infolge page faults die CPU durch Systemprogramme für eine bestimmte Systemverwaltungszeit (system overhead) belegt ist.

Das Ablaufgeschehen wird nicht unwesentlich durch die statistisch schwankenden Bedienungszeiten in CPU und DTU bestimmt. Messungen an realen Systemen haben gezeigt, daß die CPU-Belegungszeiten zwischen zwei page faults (Benutzerprogramme) zu hyperexponentiellem Charakter neigen. Demgegenüber sind Zugriffszeiten auf rotierende Speicher hypoexponentieller Natur. Bild 5 zeigt typische Verläufe der Wahrscheinlichkeits-VF für die Zufallsvariablen T_{H1} (CPU-Belegungszeit) und T_{H2} (DTU-Belegungszeit) im Vergleich zur exponentiellen Verteilungsfunktion. Die starke Streuung der CPU-Belegungszeiten zwischen zwei page faults rührt von dem Lokalitätsverhalten der Programme her: es treten bevorzugt sehr kurze Zeiten (bei Lokalitätswechsel) und sehr lange Zeiten (Verweil-

dauer innerhalb einer Lokalität) auf. Die schwache Streuung der DTU-Belegungszeiten hat ihre Ursachen in einer konstanten Übertragungszeit pro Seite und einer nahezu linearen Speicherzugriffszeit (Positionierzeit bei rotierenden Speichern); die VF wird ferner beeinflußt durch Strategien, welche die Reihenfolge der Abfertigung wartender DTU-Anforderungen entsprechend der momentanen Position der Leseköpfe verändern.

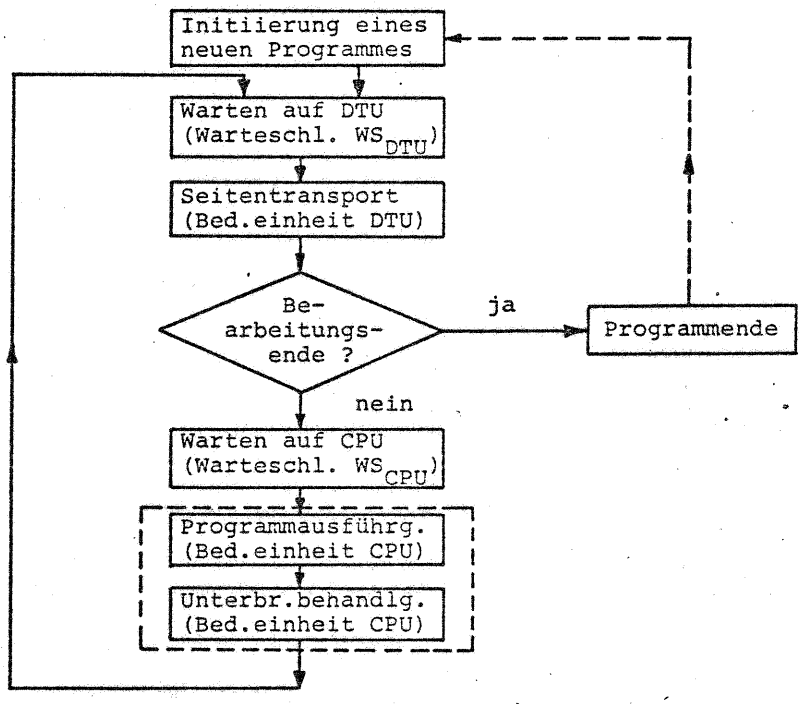

Bild 4. Zustandsfolge eines Programmes entsprechend Bild 3

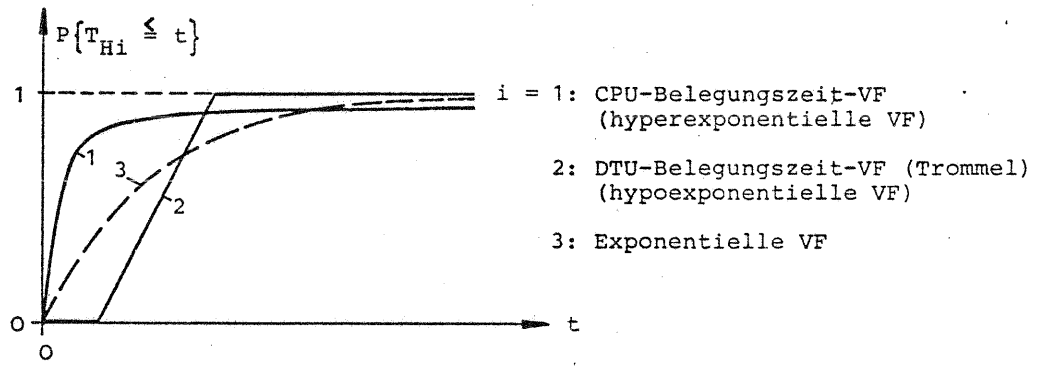

Bild 5. Belegungszeit-Verteilungsfunktionen für CPU und DTU

Zur analytischen Untersuchung werden Bedienungszeit-VF zugrundegelegt, welche in den ersten beiden Momenten (Mittelwert und Varianz) realistisch sind. Um das Warteschlangenmodell noch analytisch exakt analysieren zu können, werden die durch zwei Momente festgelegten Charakteristiken durch Zusammenschaltung aus fiktiven exponentiellen Teilphasen wie folgt erzeugt, vergl. Bild 6:

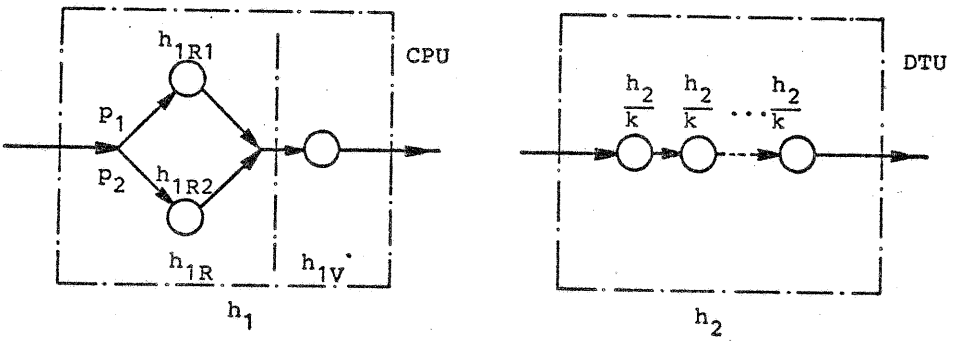

Bild 6. Ersatz-Darstellungen der Bedienungszeit-Charakteristiken für CPU und DTU

a) Bedienungszeit-Charakteristik der CPU

Jede CPU-Bedienungsphase (Mittelwert h_1) setzt sich aus zwei Teilen zusammen:

1. CPU-Bedienung für Benutzerprogramm-Bearbeitung
   Mittelwert: h_{1R}
   Verteilung: Hyperexponentiell; repräsentiert durch 2 alternative exponentielle Teilphasen mit Mittelwerten h_{1R1}, h_{1R2} und Alternativwahrscheinlichkeiten $p_1, p_2=1-p_1$ entspr. vorgegebenem Variationskoeffizienten c_{1R}.
2. CPU-Bedienung für Unterbrechungsbehandlung (Systemprogramme)
   Mittelwert: h_{1V}
   Verteilung: Exponentiell.

b) Bedienungszeit-Charakteristik der DTU

Die Approximation von Trommel- und Plattenzugriffen unter Berücksichtigung der ersten beiden Momente ergibt [10]:

Mittelwert: h_2
Verteilung: Hypoexponentiell, repräsentiert durch eine Erlang-k-VF, welche sich aus k seriellen exponentiellen Teilphasen mit Mittelwert h_2/k erzeugen läßt.

Bemerkung: Mit der Erlang-k-VF lassen sich nur diskrete Variationskoeffizienten $c_2=1/\sqrt{k}$ realisieren; ist eine genauere Approximation erforderlich, so kann mit einer seriellen Anordnung aus Erlang-k-VF und einer weiteren exponentiellen VF jeder beliebige Variationskoeffizient $1/\sqrt{k} \geq c_2 \geq 1/\sqrt{k+1}$ eingestellt werden, $k = 1,2,\ldots$

3. MODELLANALYSE

Die Analyse des Warteschlangenmodells nach 2.4 erfolgt im Zusammenhang mit den übrigen Voraussetzungen über Betriebssystem-Strategien (2.2) und Programmeigenschaften (2.3).

3.1 Parameterfestlegung

In dem Warteschlangenmodell nach Bild 3 und Bild 6 sind folgende Parameter festzulegen:

M	Multiprogramminggrad
h_{1R}	Mittelwert der CPU-Benutzerprogramm-Bearbeitungsphasen
h_{1R1}, h_{1R2} p_1, p_2	Parameter der hyperexponentiellen Ersatz-Darstellung der CPU-Benutzerprogramm-Bearbeitungsphasen. Festlegung durch h_{1R} und den Variationskoeffizienten c_{1R}
h_{1V}	Mittelwert der CPU-Systemprogrammphasen
h_2	Mittelwert der DTU-Bedienung
k	Phasenzahl der Erlang-k-VF für DTU-Bedienungszeiten
p_o	Verzweigungswahrscheinlichkeit

Die Parameter werden wie folgt festgelegt:

a) Aus der Vorgabe von Multiprogramminggrad M, Arbeitsspeichergröße S, Programmgröße L sowie der Ersetzungsstrategie und dem Programmtyp folgt zunächst $x = S/(ML)$ und damit aus Bild 2 die Fehlseitenrate f.

b) Mit der weiteren Vorgabe der mittleren CPU-Gesamt-Ausführungsdauer h_{GR} lassen sich folgende Bilanzen aufstellen, wobei der Wert $1/p_o - 1$ die aus Bild 3 folgende mittlere Anzahl von CPU-Bearbeitungsphasen eines Programmes darstellt:

$$h_{GR} = \left(\frac{1}{p_o} - 1\right) \cdot h_{1R} \tag{1}$$

$$h_{1Ref} = f \cdot h_{1R} \quad , \tag{2}$$

woraus folgt

$$h_{1R} = \frac{h_{1Ref}}{f} \tag{3}$$

$$p_o = \frac{1}{1 + f \cdot \frac{h_{GR}}{h_{1Ref}}} \quad . \tag{4}$$

c) Aus h_{1R} nach (3) und c_{1R} aus bekannten Messungen folgen

$$h_{1R\,1,2} = \frac{h_{1R}}{1 \pm \sqrt{1 - 2/(1+c_{1R}^2)}} \tag{5}$$

$$p_{1,2} = \frac{h_{1R}}{2 \cdot h_{1R1,2}} \quad . \tag{6}$$

d) Die Parameter h_{1V} und h_2 sind durch das Rechnersystem selbst festgelegt. Für k wurden die Werte 3 (Trommel- bzw. Festkopfplattenzugriffe) bzw. 5 (Plattenzugriffe) ermittelt.

3.2 Analyse des Warteschlangenmodells

a) Bekannte Lösungen

Eine allgemeine Lösung für geschlossene Warteschlangennetze existiert nur im Falle exponentieller Bedienungsdauern [7]. Unter allgemeineren Voraussetzungen lassen sich i.a. nur Näherungslösungen ableiten [8]. Ein ähnliches Modell mit einer entartet negativ-exponentiellen CPU-Bedienungszeit-Charakteristik, exponentiell verteilten DTU-Bedienungszeiten und einer Durchsatz-optimalen CPU-Zuteilungsstrategie hat Walke [9] analysiert, wobei ebenfalls gemessene Fehlseiten-Charakteristiken zugrundegelegt wurden. Es werden dort Aussagen über den optimalen Multiprogramminggrad gemacht, welche hier prinzipiell bestätigt werden. Die Analyse in [9] macht in Erweiterung dazu noch Aussagen über den Einfluß streuender Programmgröße.

b) Analyse durch Phasenmethode

Da die Bedienungszeiten der CPU- und DTU-Stufe nach Bild 6 aus fiktiven exponentiellen Phasen zusammengesetzt wurden, kann eine exakte Analyse mit Hilfe eines geeignet definierten mehrdimensionalen Markoff-Prozesses erfolgen nach der sog. Phasenmethode [10,11]. Als Systemzustand wird definiert:

$$(x_1, x_{1P}, x_{2P})$$

wobei

x_1 = Anzahl der Anforderungen in CPU und WS_{CPU}, $x_1 = 0,1,\ldots,M$

x_{1P} = Phasenzustand der CPU-bedienten Anforderung, wobei

$$x_{1P} = \begin{cases} 1 & \text{Benutzerprogrammphase mit Mittelwert } h_{1R1} \\ 2 & \text{Benutzerprogrammphase mit Mittelwert } h_{1R2} \\ 3 & \text{Systemprogrammphase mit Mittelwert } h_{1V} \end{cases}$$

x_{2P} = Phasenzustand der DTU-bedienten Anforderung, $x_{2P} = 1,2,\ldots,k$.

Die Zustände $x_{1P} = 0$ bzw. $x_{2P} = 0$ bedeuten, daß die Bedienungseinheiten CPU bzw. DTU nicht belegt sind.

Das Zustandsgleichungssystem wird mit Hilfe eines iterativen Verfahrens (sukzessive Überrelaxation) aufgelöst. Aus den Zustandswahrscheinlichkeiten werden u.a. folgende Kenngrößen gewonnen:

$Y_{CPU,R}$	Auslastung der CPU durch Benutzerprogramme
$Y_{CPU,V}$	Auslastung der CPU durch Systemprogramme
D	Durchsatz (Anzahl fertig bearbeiteter Programme je Zeiteinh.)
Y_{DTU}	Auslastung der DTU
Ω_{CPU}	Mittlere Warteschlangenlänge von bearbeitbaren Programmen
Ω_{DTU}	Mittlere Warteschlangenlänge von Programmen, welche auf Seitennachladung warten
t_F	Mittlere Durchlaufzeit eines Programms.

4. ERGEBNISSE

4.1 Voraussetzungen zu den numerischen Ergebnissen

Das Warteschlangenmodell nach Bild 3 und Bild 6 wurde für die sechs verschiedenen Programmtypen nach Bild 2 untersucht, wobei gewählt wurde:

S	= 50	c_{1R}	= 2
L	= 50, 100	h_{1V}	= 10 msec
h_{GR}	= 100 sec	h_2	= 25 msec
h_{1Ref}	= 10 μsec	k	= 3 (Trommel)

4.2 Optimale CPU-Auslastung und mittlere Durchlaufzeiten

In Bild 7 bzw. Bild 8 sind die CPU-Auslastungen $Y_{CPU,R}$ in Abhängigkeit des Multiprogramminggrades M für die sechs Programmtypen nach Bild 2 unter den Annahmen L = S bzw. L = 2S aufgetragen. Man erkennt, daß die Programmtypen a und d, welche eine schlechte Lokalität besitzen oder deren Seiten durch Fixierung an den Arbeitsspeicher gebunden werden, für M > 1 nicht geeignet sind. Dagegen weisen die Typen b,c,e und f Eigenschaften auf, welche die Wahl eines Durchsatz-optimalen Multiprogramminggrades M $\geq$1 nahelegen.

Wie z.B. für den Programmtyp f deutlich wird, existiert ein optimaler Multiprogramminggrad M = 3 bzw. M = 2; unterhalb dieser Werte sinkt die CPU-Auslastung infolge Unterbeschäftigung der CPU (DTU ist Engpaß),oberhalb davon sinkt die CPU-Auslastung infolge thrashing, welches einen stark ansteigenden Anteil der CPU-Auslastung infolge Systemverwaltungszeiten, $Y_{CPU,V}$, bedingt (vergl. Bild 7 und Bild 8).

Ein Vergleich der Bilder 7 und 8 mit Bild 2 hinsichtlich der Lage des Durchsatz-Optimums läßt folgenden Schluß zu, daß der optimale Multiprogramminggrad offenbar mit der Existenz und Lage eines ausgeprägten "Knickes" in der Fehlseiten-Charakteristik zusammenhängt ("Paracore"x_o). Der Kehrwert $1/x_o$ bestimmt den optimalen Multiprogramminggrad

$$M_{opt} \cong \frac{S}{L \cdot x_o} \quad . \tag{7}$$

Diese Ergebnisse können dazu dienen, um mit Hilfe von dynamisch gemessenen Fehlseiten-Charakteristiken eine optimale Steuerung des Multiprogramming durchzuführen. Ferner können solche Untersuchungen dazu genutzt werden, neue Systeme richtig auszulegen bzw. Engpässe an bestehenden Systemen gezielt zu beseitigen wie bei Fragen der Arbeitsspeicher-Erweiterung oder der Kanal-Erweiterung.

In Bild 9 schließlich ist die mittlere Durchlaufzeit t_F eines Programmes in Abhängigkeit des Multiprogramminggrades M angegeben. Allgemein steigt t_F mindestens linear mit M; man bemerkt jedoch, daß die Zunahme von t_F umso geringer ist, je flacher das Maximum der CPU-Auslastung ausfällt.

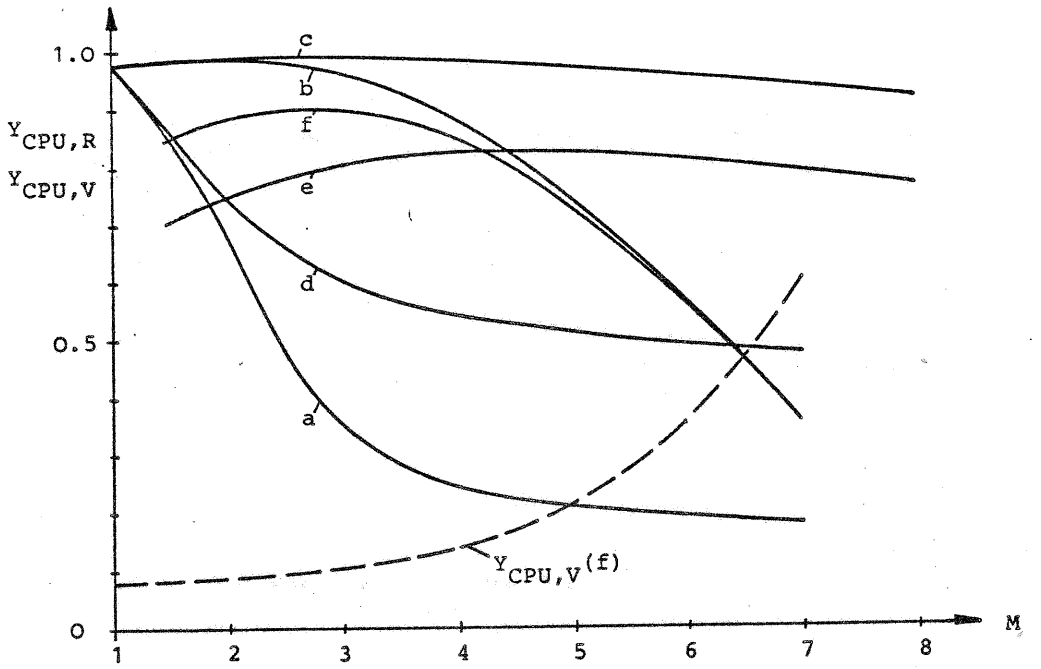

Bild 7. Auslastung der CPU in Abhängigkeit des Multiprogramminggrades
Parameter: L = S = 50.

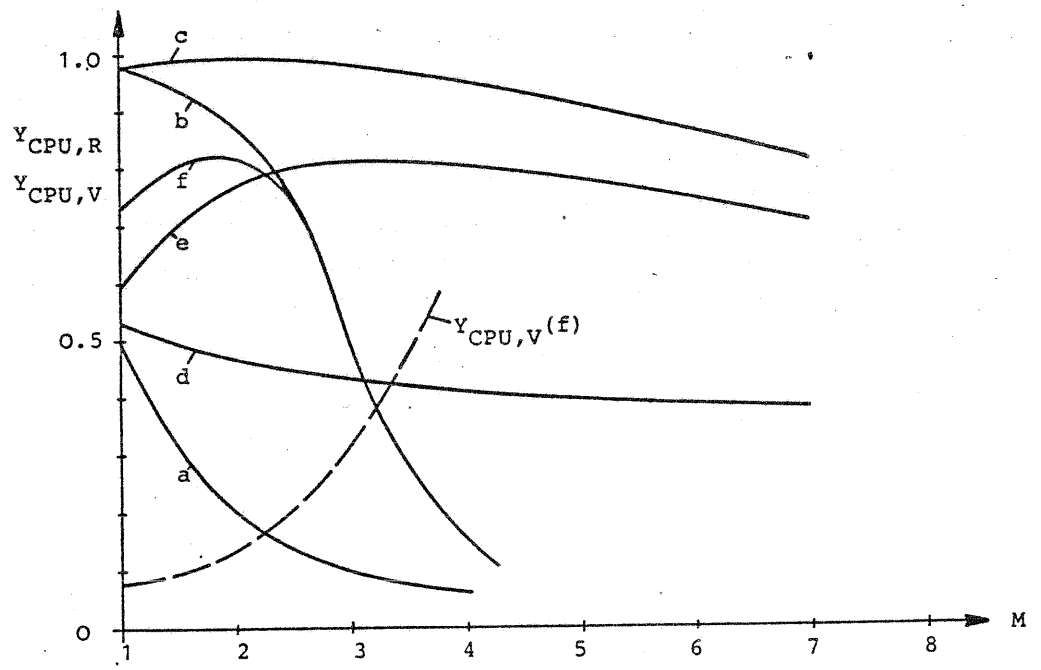

Bild 8. Auslastung der CPU in Abhängigkeit des Multiprogramminggrades
Parameter: L = 2S = 100.

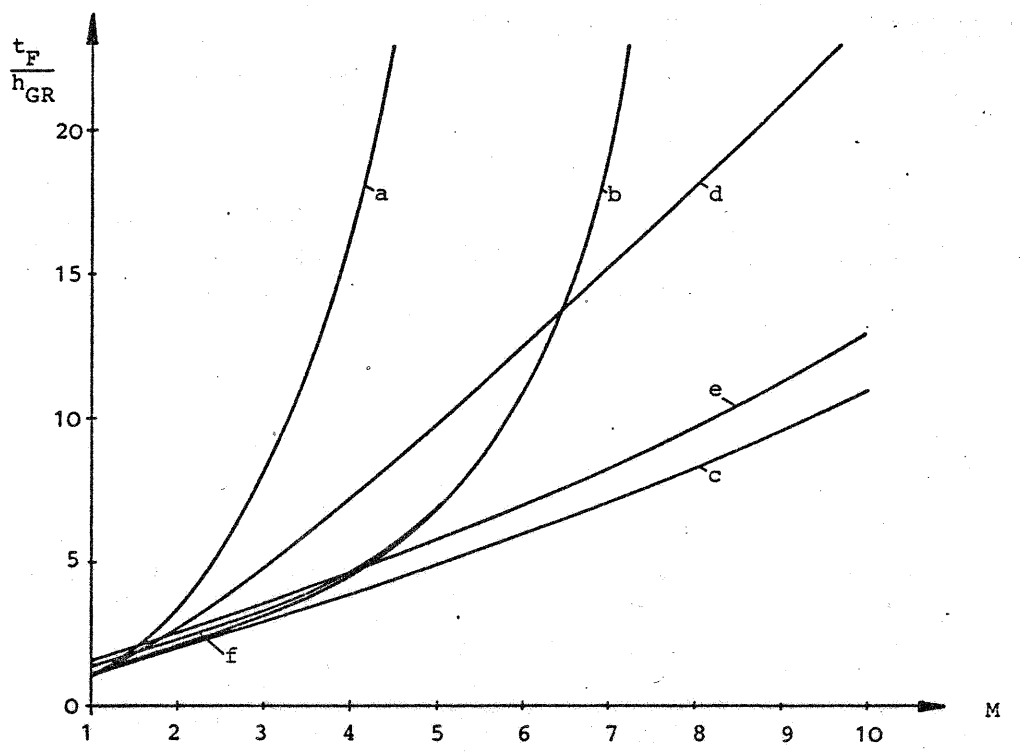

Bild 9. Bezogene mittlere Durchlaufzeit eines Programmes in Abhängigkeit des Multiprogramminggrades
Parameter: L = S = 50.

5. ERWEITERUNGEN

Aufbauend auf diesen und früheren Untersuchungen [3] wurde eine Modellhierarchie entworfen, welche u.a. den Ein-/Ausgabeverkehr von größeren Datenmengen (file I/O) über eigene Kanäle berücksichtigt. Die Untersuchungen werden auf zwei Ebenen durchgeführt:

a) Simulation

Die Simulation erfolgt für wesentlich detailliertere Modelle unter Berücksichtigung von

- mehreren DTU für page I/O
- mehreren DTU für file I/O
- künstlich erzeugten Seitenreferenzketten
- ASP-Verwaltung für die Seiten der einzelnen Programme
- Seitenersetzungsstrategien (LRU, Working Set)
- Suspendierung ineffektiv laufender Programme
- konstantem sowie variablem Multiprogramminggrad
- Systemverwaltungszeiten.

b) Mathematische Analyse

Hierfür wurden komplexere Warteschlangenmodelle entworfen, welche nach einem approximativen Verfahren analysiert werden [11,12].

ZUSAMMENFASSUNG

In der vorliegenden Untersuchung wurde ein Warteschlangenmodell für Rechnersysteme mit zweistufiger Speicherhierarchie, virtuellem Speicherprinzip und Paging unter Einbeziehung realer Programmeigenschaften und Betriebssystem-Strategien exakt analysiert. Es wurde unter vereinfachenden Voraussetzungen gezeigt, daß der Durchsatz-optimale Multiprogrammingrad mit Hilfe solcher Untersuchungen bestimmt werden kann in Abhängigkeit von Lokalitätseigenschaft realer Benutzerprogramme und Systemparametern.

SCHRIFTTUMSVERZEICHNIS

[1] Coffman,E.G., Denning,P.J.: Operating Systems Theory. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[2] Denning,P.J., Graham,G.S.: Multiprogrammed Memory Management. IEEE Proc. on Interactive Computer Systems (to appear).

[3] Herzog,U., Krämer,W., Kühn,P., Wizgall,M.: Analyse von Betriebssystem-Modellen für Rechnersysteme mit Multiprogramming und Paging. GI-NTG Fachtagung "Struktur und Betrieb von Rechensystemen", Braunschweig, 20.-22.3.1974. Lecture Notes in Computer Science, Springer-Verlag, Berlin/Heidelberg/New York 1974, S.266-288.

[4] Oliver,N., Chu,W.W., Opderbeck,H.: Measurement Data on the Working Set Replacement Algorithm and their Applications. Proc. Symp. on Computer-Communications Networks and Teletraffic, Brooklyn, 4.-6.4.1972. Polytechnic Press of the PIB, S.113-124.

[5] Hatfield, D.J.: Experiments on Page Size, Program Access Patterns, and Virtual Memory Performance. IBM J. Res. and Develop. 16 (1972), S. 58 - 66.

[6] Wolf,P.: Eine Methode zur Untersuchung von Programmen bezüglich eines Betriebssystems mit virtuellem Speicher - Anwendung zur Vorhersage des Programmverhaltens. GI-NTG Fachtagung "Struktur und Betrieb von Rechensystemen", Braunschweig, 20.-22.3.1974. Lecture Notes in Computer Science, Springer-Verlag, Berlin/Heidelberg/ New York 1974, S. 289 - 300.

[7] Gordon,W.J., Newell,G.F.: Closed Queuing Systems with Exponential Servers. Opns. Res. 15 (1967), S. 254 - 265.

[8] Chandy,K.M., Herzog,U., Woo,L.: Approximate Analysis of General Queuing Networks. IBM J. Res. and Develop.19(1975), S. 43 - 49.

[9] Walke,B.: Durchsatzberechnung für Rechenanlagen bei wählbarer Aufteilung des Arbeitsspeichers unter mehrere Programme unterschiedlichen Platzbedarfs. Dissertation Univ. Stuttgart, 1975.

[10] Cox,D.R.: A Use of Complex Probabilities in the Theory of Stochastic Processes. Proc. Camb. Phil. Soc. 51(1955), S. 313 - 319.

[11] Ertelt,R., Kühn,P.: Analyse komplexer Warteschlangennetze für Rechnersysteme. Monographie Institut für Nachrichtenvermittlung und Datenverarbeitung, Univ. Stuttgart, 1975.

[12] Kühn,P.: Analysis of Complex Queuing Networks by Decomposition (Veröffentlichung in Vorbereitung).