KLASSIFIZIERUNG UND ANALYSE VON VERKEHRSMODELLEN FÜR DAS ABLAUFGESCHEHEN IN RECHNERSYSTEMEN

Von Ulrich Herzog, Paul Kühn und Albrecht Zeh

Mitteilung aus dem Institut für Nachrichtenvermittlung und Datenverarbeitung
Universität Stuttgart

1. EINLEITUNG

Die Vielfalt der Probleme, die bei modernen Rechnersystemen auftreten, kann man nur dann befriedigend lösen, wenn bei der Entwicklung von Hardware- und Softwarestrukturen verkehrsmäßige Gesichtspunkte berücksichtigt werden. Verkehrsmäßige Untersuchungen ermöglichen eine Bestimmung maschineninterner Engpässe sowie Angaben über die Gesamtleistungsfähigkeit eines Rechnersystems.

In letzter Zeit ist eine Vielzahl von Teiluntersuchungen aus den verschiedenartigsten Anwendungsbereichen in der Literatur bekannt geworden. In diesem Beitrag wird nunmehr versucht, eine Klassifizierung der Verkehrsmodelle vorzunehmen. Für ein spezielles Realzeitsystem wird eine mathematische Analyse durchgeführt.

Zunächst wird ein kurzer Überblick gegeben über die mathematische Beschreibung der Verkehre, den Einfluß von Systemstrukturen und die Definition wichtiger charakteristischer Verkehrsgrößen. Es werden ferner die wichtigsten Methoden (rechnerische Verfahren und Simulation) zur Bestimmung dieser Verkehrsgrößen kurz erläutert.

Im Hauptteil der Arbeit werden verschiedene Verkehrsmodelle beschrieben und klassifiziert. Eine Unterteilung in Modelle für den Prozessorverkehr, den Speicherverkehr sowie Modelle für den Verkehr zwischen verschiedenartigen Funktionseinheiten hat sich als sinnvoll erwiesen. Es wird versucht, wesentliche Merkmale herauszuarbeiten, um dadurch den gemeinsamen Charakter von Modellen aus den verschiedenen Anwendungsbereichen aufzuzeigen. Wegen der stürmischen Entwicklung neuer Hardware- und Softwarestrukturen kann die hier vorliegende Einteilung nur der Beginn einer systematischen Klassifizierung sein.

Im vierten Kapitel wird ein spezielles Prozessormodell mit zwei Typen von Anforderungen (Realzeit- und Hintergrundprogramme) analytisch behandelt. Realzeitanforderungen haben hierbei gegenüber Hintergrundprogrammen unterbrechende Priorität, wobei die Unterbrechungsstrategie selbst noch zeitabhängig ist. Es wird gezeigt, wie mit Hilfe dieser Strategie die Anzahl der Unterbrechungen unter Einhaltung der Realzeitforderungen optimiert werden kann.

2. MATHEMATISCHE BEHANDLUNG VON VERKEHRSABLÄUFEN

2.1 Allgemeines

Das Ablaufgeschehen innerhalb von Rechnersystemen wird durch den Transport von Befehlen und Daten zwischen Teilsystemen, sowie deren Bearbeitung in den verschiedenen Systemeinheiten bestimmt. Durch hardwaremäßige Engpässe oder eine ungünstige Konzeption des Betriebssystems treten Warte- und Blockierungszeiten auf, welche einen reibungslosen Betrieb behindern.

Der zeitliche Ablauf derartiger Transport-, Warte- und Bearbeitungsphasen entspricht einem speziellen stochastischen Prozess, welcher mit Methoden der Wahrscheinlichkeitstheorie beschrieben werden kann. Das gesamte System kann durch folgende Kriterien vollständig gekennzeichnet werden:

- Ankunftsprozess: Die Verteilung der Ankunftsabstände der Befehle bzw. Daten.
  Die Verteilungsfunktion A(t) ist definiert als die Wahrscheinlichkeit, daß der Ankunftsabstand a höchstens gleich der beliebigen Zeit t ist

$$A(t) = P\{a \leq t\} . \qquad (2.1)$$

Beitrag zur NTG/GI-Fachtagung "Rechner- und Betriebssysteme: Analyse, Simulation und Entwurf", Darmstadt, 12.-14.4.1972. Nachrichtentechn. Fachber. Bd. 44 (1972), S. 181 - 198.

Bezeichnet man den mittleren Ankunftsabstand mit a_m, so wird der Kehrwert als mittlere Ankunftsrate definiert:

$$\lambda = 1/a_m. \tag{2.2}$$

- Endeprozess: Verteilung der Bedienungs- bzw. Transportzeiten b. Es gilt entsprechend Gl.(2.1) und (2.2)

$$B(t) = P\{b \leq t\} \tag{2.3}$$

$$\mu = 1/b_m. \tag{2.4}$$

- Systemeinfluß: Der Einfluß des Systems ist durch die Systemstruktur und die Art der Abfertigung ankommender und wartender Aufgaben festgelegt.

Ziel der verkehrstheoretischen Untersuchungen ist es, mit Hilfe dieser Angaben charakteristische Gütekriterien abzuleiten, um damit die Effektivität eines Rechnersystems beurteilen bzw. verbessern zu können. Für die Untersuchung des Verkehrsverhaltens gibt es rechnerische Verfahren und die Simulation.

2.2 Wichtige Ankunfts- und Endeprozesse (vergl. z.B. Ferschl[1], Syski[2])

Die am häufigsten für Ankunfts- und Endeprozesse benutzte Verteilungsfunktion ist die negativ-exponentielle Verteilung:

$$P\{x \leq t\} = 1 - \exp(-t/x_m) . \tag{2.5}$$

mit x_m als Mittelwert der Verteilung von x. In diesem Fall hat man einen sog. Markoff'schen Prozess (vergl. auch Abschn.2.5), welcher mathematisch relativ einfach zu behandeln ist. Endeprozesse werden jedoch häufig besser durch eine konstante Verteilung approximiert:

$$P\{x \leq t\} = \begin{cases} 0 & \text{für } 0 \leq t < x_m \\ 1 & \text{für } t \geq x_m \end{cases} . \tag{2.6}$$

Die mathematische Behandlung derartiger Verkehre ist jedoch bedeutend schwieriger, da die sog. Markoff'sche Eigenschaft (Unabhängigkeit von der Vorgeschichte) verlorengeht. Beide Verteilungen, Gl.(2.5) und (2.6), sind Grenzwerte der Erlang-k-Verteilungsfunktion:

$$P\{x \leq t\} = 1 - \exp(-k\cdot\frac{t}{x_m})\cdot\sum_{\xi=0}^{k-1}\frac{(k\cdot\frac{t}{x_m})^{\xi}}{\xi!} \quad , t \geq 0. \tag{2.7}$$

Diese Verteilung wird erfolgreich angewendet, um Ankunfts- und Endeprozesse zu beschreiben, deren Verteilungsfunktion zwischen konstanter und negativ-exponentieller liegen.

Eine Familie von Verteilungsfunktionen ohne Begrenzung ist die gemischte Erlang-Verteilung, bestehend aus einer Summe gewichteter Erlang-Verteilungen. Der einfachste Fall ist die sog. hyperexponentielle Verteilungsfunktion 2. Ordnung:

$$P\{x \leq t\} = 1 - p\cdot\exp\left[-2p\cdot\frac{t}{x_m}\right] - (1-p)\cdot\exp\left[-2(1-p)\cdot\frac{t}{x_m}\right] \quad , t \geq 0. \tag{2.8}$$

Erlang-k-Verteilung und hyperexponentielle Verteilung erlauben eine gute näherungsweise Beschreibung vieler in der Praxis auftretender Ankunfts- und Endeprozesse; die mathematische Behandlung ist jedoch häufig recht schwierig.

Ein elegantes Verfahren besteht darin, beliebige Typen von Verteilungsfunktionen durch stückweise exponentielle Verteilungsfunktionen anzunähern (Marte[3]); in den einzelnen Phasen sind dann die Markoff'schen Bedingungen erfüllt. Der einfachste Fall sind zwei an einem Grenzzeitpunkt t_g zusammengesetzte Phasen:

$$P\{x \leq t\} = \begin{cases} 1 - \exp(-\frac{t}{x_{m_1}}) & \text{für } t \leq t_g \\ 1 - \exp(-\frac{t_g}{x_{m_1}})\cdot\exp(-\frac{t-t_g}{x_{m_2}}) & \text{für } t \geq t_g \end{cases} . \tag{2.9}$$

2.3 Der Systemeinfluß

Transport- und Bedienungsprozesse werden sowohl durch die Systemstruktur als auch durch die Art der Abfertigung eintreffender oder wartender Anforderungen beeinflußt. Die Strukturmodelle wer-

den im 3. Kapitel eingehend behandelt. Um im Folgenden Wiederholungen zu vermeiden, seien hier die wichtigsten Abfertigungsstrategien zusammengestellt und kurz erläutert; auf spezielle Verfahren wird an den entsprechenden Stellen hingewiesen.

2.3.1 Abfertigung ohne Prioritäten

Die bekanntesten Abfertigungsstrategien für gleichberechtigte Anforderungen sind folgende (siehe z.B. Ferschl[1], Syski[2]):

- FIFO first-in, first-out, d.h. die zuerst eingetroffene Anforderung wird zuerst bedient.
- LIFO last-in, first -out, d.h. die zuletzt eingetroffene Anforderung wird zuerst bedient.
- RANDOM die wartenden Anforderungen werden zufällig ausgesucht und bedient.

Sind mehrere Warteschlangen mit gleichberechtigten Anforderungen vorhanden, so kommen zu diesen Abfertigungsstrategien noch verschiedene Möglichkeiten zur Auswahl einer bestimmten Warteschlange hinzu (sog. interqueue-discipline, vergl. Kühn[4]).

2.3.2 Abfertigung mit Prioritäten

Bei Prioritäten unterscheidet man interne und externe Prioritätsklassen. Interne Prioritäten werden bei an sich gleichrangigen Aufgaben gesetzt, um z.B. einen maximalen Durchsatz zu erzielen. Beispiele hierfür sind die SJF-Strategie (shortest-job-first-served, d.h. die Anforderung mit der kürzesten Bedienungszeit wird zuerst bedient), mathematisch behandelt von Phipps[5], oder die Abfertigungsstrategie von Marte[3]. Externe Prioritäten geben Auskunft über die Wichtigkeit einer Anforderung im Vergleich mit anderen konkurrierenden Anforderungen. Sie sind oft fest vorgegeben für bestimmte Anforderungen, können aber auch wartezeitabhängig sein.

Sowohl interne als auch externe Prioritäten können rein unterbrechend, rein nichtunterbrechend oder eine Mischform zwischen beiden Extremen sein. Rein unterbrechend bedeutet, daß die Bearbeitung einer Anforderung beim Eintreffen einer Anforderung höherer Priorität sofort unterbrochen wird (mathematische Behandlung siehe z.B. Stephan[6], Gaver[7]). Bei den rein nichtunterbrechenden Strategien werden zwar Anforderungen der wichtigsten Prioritätsklasse vor allen anderen Wartenden bedient, eine momentane Bearbeitungsperiode kann aber nicht unterbrochen werden (siehe z.B. Cobham[8], Takács[9]). Bei gemischten Prioritäten unterscheidet man zeitabhängige Unterbrechungsstrategien (siehe z.B. Coffman[10] und Kapitel 4 dieser Arbeit) und Unterbrechungsdistanz-Prioritäten (beliebige Folgen unterbrechender und nichtunterbrechender Klassen, Herzog [11]).

Eine sehr ausführliche Zusammenstellung und Behandlung von Prioritätssystemen geben z.B. Gaver [12] und Jaiswal [13].

2.4 Charakteristische Größen

Die wichtigsten charakteristischen Größen sind mittlerer bzw. maximaler Durchsatz (mittlere bzw. maximale Anzahl von Anforderungen, welche pro Zeiteinheit transportiert oder bearbeitet werden können), die Anzahl der Unterbrechungen pro Zeiteinheit bzw. pro Bedienungsdauer einer individuellen Anforderung, die Wartezeitverteilungsfunktion (Wahrscheinlichkeit, daß eine Anforderung höchstens eine bestimmte Zeit warten muß) und die daraus ableitbaren Größen wie z.B. die Wartewahrscheinlichkeit (Wahrscheinlichkeit, daß eine eintreffende Anforderung überhaupt warten muß), die mittlere Wartezeit (bezogen auf alle Anforderungen oder nur auf jene, die auch tatsächlich warten müssen), die mittlere Antwortzeit (Summe aus den Mittelwerten von Warte- und Bedienungszeit) sowie die mittlere Warteschlangenlänge. Bei Systemen mit begrenztem Speicherraum ist noch die sog. Verlust- oder Überlaufwahrscheinlichkeit (Wahrscheinlichkeit, daß eine eintreffende Anforderung keinen freien Speicherplatz findet und deshalb "verlorengeht") von Interesse.

2.5 Berechnungsmethoden

Aus Platzgründen kann nur ein knapp gefaßter Überblick über die wichtigsten Berechnungsmethoden gegeben werden. Bücher von Ferschl[1], Syski[2], Morse[14], Takács[15], Saaty[16] und vielen anderen Autoren ermöglichen ein ausführliches Studium von Wartezeitproblemen und deren mathematische Behandlung. Zur Berechnung unterscheiden wir hier zwei große Gruppen von Prozessen, sog. Markoff'sche und Nicht-Markoff'sche Prozesse.

2.5.1 Markoff'sche Prozesse

Charakteristikum der Markoff'schen Prozesse ist die Unabhängigkeit des Prozessverlaufs ϕ von der Vorgeschichte, d.h. für die bedingten Zustandswahrscheinlichkeiten zu einem beliebigen Zeitpunkt t_{n+1} gilt:

$$P\left\{\phi_{t_{n+1}} = j \,\middle|\, \phi_{t_o} = x_o, \phi_{t_1} = x_1, \ldots, \phi_{t_n} = x_n\right\} = P\left\{\phi_{t_{n+1}} = j \,\middle|\, \phi_{t_n} = x_n\right\}, \qquad (2.10)$$

$$t_o < t_1 < t_2 < \ldots$$
$$n = 0,1,2,\ldots$$

Dieses Verhalten kann für alle Markoff'schen Prozesse mit Hilfe des sog. Chapman-Kolmogoroff'schen Gleichungssystems beschrieben werden. Durch das Auflösen des Gleichungssystems können die Zustandswahrscheinlichkeiten sowie die charakteristischen Größen bestimmt werden. Markoff'sche Prozesse haben den großen Vorteil, daß man stets denselben Lösungsweg einschlagen kann. Schwierigkeiten - teilweise allerdings erhebliche - treten vor allem erst bei der Auflösung bzw. Auswertung der linearen Gleichungs- bzw. Differentialgleichungssysteme auf.

2.5.2 Nicht-Markoff'sche Prozesse

Nicht-Markoff'sche Prozesse sind alle jene Prozesse, bei welchen der Prozessverlauf - mindestens für eine gewisse Zeit - von der Vorgeschichte abhängig ist. Ein einheitlicher Lösungsansatz wie bei den Markoff'schen Systemen ist nicht mehr möglich. Dennoch versucht man durch einen Trick, sich die Vorteile, die sich bei der Behandlung Markoff'scher Systeme ergeben, zunutze zu machen: man ersetzt das zeitlich abhängige Verhalten eines eindimensionalen Nicht-Markoff'schen Prozesses durch das Verweilen in fiktiven exponentiellen Phasen, d.h. durch einen mehrdimensionalen Markoff-Prozess (Phasenmethode). Eine zweite Möglichkeit ist das Verfahren der Eingebetteten Markoff-Kette: man betrachtet einen beliebigen stochastischen Prozess nur zu solchen diskreten Zeitpunkten, bei denen die Zustände die Markoff-Eigenschaften besitzen (vergl. auch Kapitel 4).

Beide Methoden sind dazu geeignet, zeitabhängige Prozesse zu beschreiben, also auch z.B. das Einschwingverhalten eines Systems zu untersuchen. Diese allgemeine Anwendbarkeit bringt zwangsläufig den Nachteil relativ umfangreicher Gleichungssysteme mit sich. Oft sind deshalb zwar prinzipiell Lösungsansätze möglich, eine numerische Auswertung scheitert aber selbst auf den größten Datenverarbeitungsanlagen. Bei Prozessen, die bereits zu Beginn der Beobachtung im eingeschwungenen Zustand sind (sog. stationäre Prozesse) erlaubt die sog. Momentenmethode mit relativ geringem Aufwand die Bestimmung des ersten Moments (z.B. der Wartezeitverteilungsfunktion), teilweise auch höherer Momente. Man verfolgt hierbei das Schicksal einer individuellen Anforderung vom Eintreffen in das System bis zum Verlassen mit allen Möglichkeiten für Unterbrechung, Zurückschieben im Speicher usw. Durch das Bilden von Erwartungswerten ist dann oft eine Bestimmung der Momente möglich.

2.6 Simulation

Bei der Simulation von Datenflüssen auf Digitalrechnern bildet man das zu untersuchende System und den Verkehrsfluß mit Hilfe eines Programms im Rechner nach. Man entwirft ein Modell des realen Systems und der tatsächlichen Verkehrsabläufe und kann damit - zeitlich gerafft gegenüber der Wirklichkeit - die Leistungsfähigkeit des Systems untersuchen.

Der Hauptvorteil der Simulation gegenüber den rein rechnerischen Verfahren liegt darin, daß man neue Probleme, deren Lösung noch nicht bekannt ist, in relativ kurzen und gut abschätzbaren Zeiträumen untersuchen kann, während ein Erfolg bei den rein rechnerischen Verfahren nicht immer gewährleistet ist. Demgegenüber besitzt die Simulation die Nachteile, daß i.a. große Rechenzeiten erforderlich sind und daß der Einfluß der einzelnen Systemparameter schwierig festzustellen ist, da für jede neue Parameterkombination ein neuer Simulationslauf erforderlich ist. Schließlich läßt sich noch feststellen, daß die Simulation wenig geeignet ist, um Optimierungen durchzuführen.

Eine Zusammenfassung vieler bei der Simulation von Rechnersystemen auftretender Probleme gibt Kümmerle [17]. Gordon [18], Kampe [19] und andere geben einen Überblick über die zahlreichen Simulationssprachen, welche eine relativ einfache und schnelle Programmierung erlauben.

3. KLASSIFIZIERUNG VON VERKEHRSMODELLEN FÜR DAS ABLAUFGESCHEHEN IN RECHNERSYSTEMEN

3.1 Allgemeines

Der Aufbau moderner Datenverarbeitungsanlagen ist so komplex, daß eine genaue Untersuchung des Gesamtsystems mit den heute zur Verfügung stehenden mathematischen Hilfsmitteln in einem Schritt nicht möglich ist. Man untersucht deshalb zunächst Teilsysteme sowie die gegenseitige Beeinflussung von Teilsystemen. Diese Untersuchungen geben eine Auskunft über maschineninterne Engpässe und erlauben eine Abschätzung der Gesamtleistungsfähigkeit des Rechnersystems. Für die mathematische Behandlung dieser Teilprobleme ist es sinnvoll, entsprechend dem Aufbau von Rechnersystemen aus funktionellen Einheiten, zu unterscheiden zwischen

- Modelle für den Prozessorverkehr
- Modelle für den Speicherverkehr
- Modelle für den Verkehr zwischen verschiedenartigen Funktionseinheiten.

Diese Probleme werden in den Abschnitten 3.2 bis 3.4 behandelt.

Ein wichtiges Teilproblem großer Datenverarbeitungssysteme ist die Absicherung gegen Überlastung sowie gegen Teil- oder Totalausfall. Diese Absicherung ist u.a. dadurch möglich, daß man eine gegenseitige Aushilfe zwischen verschiedenen Datenverarbeitungssystemen vorsieht. Verschiedene Systemkonfigurationen und Strategien werden hierfür von Herzog und Kühn [20] diskutiert und im Hinblick auf ihre verkehrsmäßige Leistungsfähigkeit miteinander verglichen.

3.2 Modelle für den Prozessorverkehr

Die Vielzahl der verschiedenen Modelle, welche in der Literatur behandelt werden, können in zwei große Gruppen eingeteilt werden:

- Modelle für Standard-Prozessoren
- Modelle für rückgekoppelte Prozessoren.

Beide Modelle sowie die damit zusammenhängenden verkehrstheoretischen Untersuchungen werden in den folgenden Abschnitten 3.2.1 und 3.2.2 diskutiert, in 3.2.3 werden Modelle für gemischte Aufgaben behandelt.

3.2.1 Modelle für Standard-Prozessoren

Als Standard-Prozessoren werden alle Prozessoren definiert, bei welchen die einzelnen Anforderungen nacheinander bearbeitet werden, d.h. wir haben aus der Sicht des Prozessors eine Stapelverarbeitung (vergl. Bild 1 und 2). Notwendige Bevorzugungen erfolgen mit Hilfe interner bzw. externer Prioritäten (Definition siehe 2.3.2). Im Gegensatz zu rückgekoppelten Prozessoren für Time-Sharing-Betrieb (vergl. 3.2.2) wird die Zahl der Programmwechsel, und deshalb auch die für das Betriebssystem notwendige Prozessorzeit, auf ein Minimum beschränkt.

3.2.1.1 Prozessor mit einer Warteschlange

Alle auf Bedienung wartenden Anforderungen werden in eine Warteschlange eingereiht, vergl.Bild 1. Verschiedene Abfertigungsstrategien (Abfertigung mit internen Prioritäten) sind bekannt geworden, um bei gleicher Prozessorbelastung die mittlere Wartezeit zu reduzieren. Phipps [5] zeigt, daß bei bekannten Bedienungszeiten eintreffender Anforderungen die Gesamtwartezeit dann am kleinsten ist, wenn die kurzen Anforderungen zuerst bedient werden (Shortest-Job-First-Strategie). Anforderungen mit langer Bedienungszeit werden dann auf Kosten kurzer Anforderungen stark verzögert.

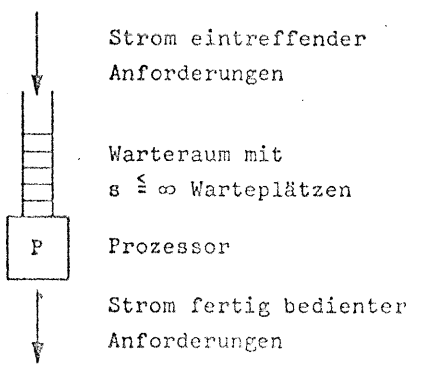

Bild 1. Standard-Prozessor mit einer Warteschlange

Hansen [21] stellt eine Strategie vor, bei welcher die internen Prioritäten proportional zur bereits verstrichenen Wartezeit und umgekehrt proportional zur (bekannten) Bedienungszeit gewählt werden. Kurze Anforderungen werden hierbei immer noch bevorzugt, allerdings nicht mehr so stark wie bei der SJF-Strategie. Häufig sind aber die für einzelne Anforderungen notwendigen Rechenzeiten nicht im voraus bekannt. Marte [3] untersucht deshalb Systeme, bei denen nur die Verteilung der Rechenzeiten bekannt ist und gibt eine

bezüglich der mittleren Wartezeit optimale Abfertigungs- und Unterbrechungsstrategie für an sich gleichrangige Anforderungen an. In einer weiteren Arbeit werden von Küspert und Marte [22] nicht nur die Rechenzeiten, sondern auch die für Programmwechsel notwendigen Verwaltungszeiten mitberücksichtigt.

Bei externen Prioritäten sind die bekanntesten Strategien "rein unterbrechende" und "rein nichtunterbrechende" Prioritäten. Exakte Ergebnisse wurden von vielen Autoren für die verschiedenartigsten Ankunfts- und Endeprozesse gefunden (vergl. 2.3.2). Oft ist es jedoch nicht sinnvoll, momentan im Prozessor laufende Programme zu jedem beliebigen Zeitpunkt zu unterbrechen. Avi-Itzhak et al [23] sowie Schrage [24] untersuchen Systeme, bei denen ein Programmwechsel von der Restbearbeitungszeit des laufenden Programmes abhängig ist. Olivier [25] ordnet jeder eintreffenden Anforderung einen Kostenwert pro Zeiteinheit des Wartens zu und bestimmt einen optimalen Rechenplan so, daß die Summe der erwarteten Gesamtkosten ein Minimum wird. Die Untersuchungen zeigen, bei welchen Typen von Verteilungsfunktionen Unterbrechungen sinnvoll bzw. nicht sinnvoll sind.

3.2.1.2 Prozessor mit mehreren Warteschlangen

Die verschiedenen auf Bedienung wartenden Anforderungen werden entsprechend ihrer Dringlichkeit in verschiedene Warteschlangen Nr.1 bis P eingeteilt; Anforderungen der höchsten Dringlichkeit bilden Klasse 1, jene der niedrigsten Klasse P, vergl. Bild 2. Nimmt man an, daß in jeder Warteschlange unbegrenzt viele Anforderungen warten können, so gelten für rein unterbrechende bzw. rein nichtunterbrechende Prioritäten die bekannten Formeln von Systemen mit einer Warteschlange. Für begrenzten Speicherraum je Warteschlange werden von Kühn [26,27] u.a. Systeme mit nichtunterbrechenden Prioritäten behandelt.

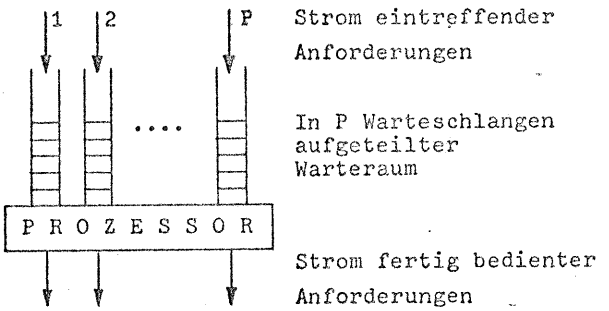

Bild 2. Standard-Prozessor mit mehreren Warteschlangen

Häufig sind für spezielle unaufschiebbare Anforderungen unterbrechende Prioritäten notwendig, andere Anforderungen sind dagegen weniger dringend und rechtfertigen deshalb keine Unterbrechung. Moderne Rechnersysteme benutzen deshalb oft sinnvolle Kombinationen aus unterbrechenden und nichtunterbrechneden Prioritäten, sog. gemischte Prioritäten, vergl.2.3.2.

Speziell für Prozessoren, welche gleichzeitig Realzeit- und (zur besseren Ausnutzung) Hintergrundaufgaben bearbeiten, sind zeitabhängige Unterbrechungsstrategien von Bedeutung. Coffman [10] untersucht eine Strategie, bei welcher eine eintreffende Realzeitanforderung eine gewisse Zeit abwartet, ob das momentan laufende Hintergrundprogramm nicht fertig wird, nach einer gewissen Geduldsdauer aber unterbricht (preemption delay). Für Realzeitanforderungen wird die mittlere Wartezeit bestimmt unter der Voraussetzung, daß stets Hintergrundprogramme zur Bearbeitung bereitstehen ("gesättigter" Hintergrund). Im Kapitel 4 wird gezeigt, daß auch für den Fall eines nicht gesättigten Hintergrunds die mittlere Wartezeit für Realzeit- und Hintergrundaufgaben bestimmt werden kann. Hängt die Abfertigung der Warteschlangen von zahlreichen verschiedenartigen Faktoren ab, die im einzelnen nicht voll übersehbar sind, so kann dieses Verhalten häufig durch eine Strategie, bei der jede Warteschlange mit einer bestimmten Wahrscheinlichkeit als nächste bedient wird, beschrieben werden (Kühn [4,26,27]).

3.2.2 Modelle für rückgekoppelte Prozessoren

Bei Time-Sharing-Systemen (siehe z.B. Kümmerle [28], Hellerman [29]) steht nicht wie bei den Standard-Prozessoren der Wunsch nach einer guten Auslastung des Prozessors im Vordergrund. Man versucht vielmehr, alle Teilnehmer "quasi-gleichzeitig" zu bedienen, um einen Dialog zu ermöglichen. Die gesamte zur Verfügung stehende Rechenzeit wird deshalb in kleine Zeitscheiben aufgeteilt und zyklisch allen auf Bedienung wartenden Anforderungen zugeteilt. Da die Zeitscheiben im Verhältnis zu den Bedienungszeiten sehr klein sind, steigt die Anzahl der Programmwechsel und damit auch die für Verwaltungsarbeiten notwendige Prozessorzeit erheblich an.

Je nach Art der Zuteilung von Zeitscheiben kann man die Anforderungen gleichberechtigt bzw. bevorzugt behandeln. In der Tendenz werden jedoch bei allen Strategien Kunden mit kurzen Programmlaufzeiten bevorzugt behandelt, d.h. sie haben kürzere Gesamtwartezeiten (vergl. auch 3.2.2.2).

Die Prozessormodelle sind teilweise so komplex, daß ihre exakte mathematische Behandlung nicht immer möglich ist. Mathematische Vereinfachungen ergeben sich, wenn man die zur Unterbrechung notwendigen Verwaltungszeiten proportional zur Zeitscheibe wählt (Kleinrock [30]) bzw. völlig vernachlässigt (Coffman, Kleinrock, Muntz, Trotter [31,32,34], Walke und Küspert [35], etc.). Diese Vereinfachung wirkt sich auf die numerischen Ergebnisse bei Strategien mit relativ wenigen Programmwechseln bzw. bei Systemen mit großen Arbeitsspeichern, in denen gleichzeitig die aktuellen Teile mehrerer Programme zur Verfügung stehen, nur gering aus. Weitere starke Vereinfachungen in der mathematischen Behandlung ergeben sich, wenn man infinitesimal kleine Zeitscheiben betrachtet; diese Ersatzmodelle sind unter dem Namen "Processor-Sharing" bekannt. Die Aussagekraft numerischer Ergebnisse dieser vereinfachten Modelle sind bezüglich der absoluten Werte kritisch zu betrachten; der Vergleich verschiedener Strategien anhand von "Processor-Sharing"-Modellen kann aber durchaus Tendenzen prinzipieller Art aufzeigen.

3.2.2.1 Round-Robin-Modelle

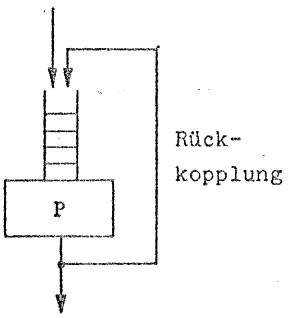

Bild 3. Round-Robin-Modell

Neu eintreffende Anforderungen werden am Schluß der Warteschlange eingeordnet. Die Abfertigung erfolgt innerhalb der Schlange in geordneter Reihenfolge (FIFO). Wird der Prozessor frei, so steht er der nächsten Aufgabe höchstens eine bestimmte Zeit q (Zeitscheibe) zur Verfügung. Benötigt die Aufgabe mehr Rechenzeit als diese Zeitscheibe, so wird sie unterbrochen und am Ende der Warteschlange wieder eingeordnet, um auf eine neue Zeitscheibe zu warten, s.Bild 3.

Zahlreiche Untersuchungen wurden für dieses Grundmodell durchgeführt, sowohl ohne (Kleinrock und Coffman [31], Kleinrock [33], Chang [36], Adiri [37], Rasch [38], usw.) als auch mit Berücksichtigung von Verwaltungszeiten (Coffman [39], Rasch [38], Adiri [37], usw.). Fast immer werden hierbei Markoff'sche Ankunfts- und Endeprozesse angenommen; Chang [36] behandelt auch den Fall beliebig verteilter Bedienungszeiten. Der Einfluß von Zeitscheiben variabler Größe wird von Rasch [38], Chang [36], Adiri [37] und Coffman [39] unter verschiedenartigen Gesichtspunkten untersucht. Der Fall infinitesimal kleiner Zeitscheiben wird u.a. von Coffman, Hsu, Kleinrock und Muntz [31-34,40] behandelt.

3.2.2.2 Feedback-Modelle

Feedback-Modelle sind Modelle für rückgekoppelte Prozessoren mit mehreren Warteschlangen. Neu eintreffende Anforderungen werden am Ende der ersten Warteschlange eingeordnet und erhalten wie alle anderen Anforderungen dieser Warteschlange zyklisch Zeitscheiben zugeteilt ("Round-Robin für Warteschlange 1"). Anforderungen, welchen bereits eine bestimmte vorgegebene Anzahl von Zeitscheiben zugeteilt worden ist (d.h. deren Bedienungszeit eine bestimmte Dauer überschreitet), werden in die zweite Warteschlange eingeordnet. Anforderungen dieser zweiten Warteschlange werden ebenfalls zyklisch Zeitscheiben zugeteilt ("Round-Robin für Warteschlange 2"), allerdings nur dann, wenn Warteschlange 1 leer ist. Ist einer Anforderung der zweiten Warteschlange wieder eine bestimmte Anzahl von Zeitscheiben zugeteilt worden und ist sie immer noch nicht fertig, so wird sie in die dritte Warteschlange eingereiht usw. Vergleicht man diese Strategie mit der reinen Round-Robin-Strategie (3.2.2.1), so erkennt man sofort, daß Anforderungen mit kurzen Rechenzeiten bei Feedback noch stärker bevorzugt werden als bei Round-Robin.

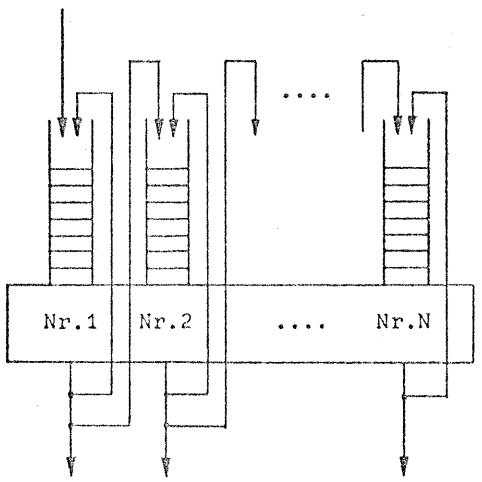

Bild 4. Feedback-Modell für einen Prozessor mit N Warteschlangen

Prioritätsminderung für spezielle Anforderungen kann man dadurch erreichen, daß man sie beim Eintreffen nicht in die erste Warteschlange, sondern sofort in eine Warteschlange höherer Ordnung einreiht (Coffman und Kleinrock [31], Adiri [41]). Verwaltungszeiten werden in den Arbeiten von Adiri [41] sowie Adiri und Avi-Itzhak [42] berücksichtigt.

Alle obengenannten Untersuchungen wurden für Ankunfts- und Endeprozesse durchgeführt, welche die Markoff'sche Eigenschaft besitzen. Nicht-Markoff'sche Bedienungszeiten werden von Walke und Küspert [35] behandelt und stückweise durch Exponentialverteilungen angenähert. Walke und Küspert untersuchen dann den Einfluß von Unterbrechungsstrategien auf die mittlere Wartezeit.

3.2.3 Prozessormodelle für gemischte Aufgaben

Um bei Time-Sharing-Systemen die Echtzeitanforderungen des Dialogbetriebs zu erfüllen, darf die Anzahl dieser Anforderungen nicht zu groß sein. Um dennoch eine hohe Auslastung des Prozessors zu erhalten, werden ihm zusätzlich Hintergrundprogramme angeboten (Hellerman [29]). Die Realzeitaufgaben werden mit einer Round-Robin- oder Feedback-Strategie abgefertigt und unterbrechen bei Bedarf die im Stapel verarbeiteten Hintergrundprogramme. Durch sinnvolle Kombination bekannter Resultate gemäß 3.2.1 und 3.2.2 kann man für verschiedene Strategien sofort die charakteristischen Verkehrsgrößen bestimmen.

3.3 Modelle für den Speicherverkehr

Die Einführung von Multiprogramming- und Time-Sharing-Betriebsweisen erfordert einen schnellen Transfer von Programmen bzw. Programmteilen und Daten. Bei diesen Betriebsweisen hat sich gezeigt, daß nicht nur der Prozessor, sondern vor allem auch die Zugriffe zum Arbeitsspeicher sowie zu peripheren Speichern den Engpaß bilden. Um Prozessoren und Speicher zeitlich gut auszunutzen, wurden hardwareseitig Konzeptionen mit Schnellspeichern sowie in Moduln unterteilte Arbeitsspeicher zur zeitlich überlappten Bearbeitung von Befehlen (pipelining, look-ahead) vorgesehen. Softwareseitig sind Konzeptionen zur Segmentierung bzw. Aufteilung von Programmen und Daten in Seiten (paging) eingeführt worden. Beim Paging-Verfahren wird nur eine begrenzte Zahl von "Programmseiten" (pages), welche i.a. verschiedenen Programmen angehören, im Arbeitsspeicher oder in einem Schnellspeicher gespeichert. Während der Bearbeitung der Programme sind daher mehrere Seitentransfers zwischen peripherem Speicher und Arbeits- bzw. Schnellspeicher erforderlich.

Die Modelle für den Speicherverkehr lassen sich in drei Kategorien einteilen:

- Modelle für den Zugriff zu peripheren Speichern
- Modelle für die Speicherplatzverwaltung innerhalb von Speichern
- Modelle für den Verkehr innerhalb einer Speicherhierarchie.

Modelle, welche im wesentlichen durch die Zusammenarbeit von Speichern mit Prozessoren gekennzeichnet sind, werden im Abschnitt 3.4 dargestellt.

3.3.1 Modelle für den Zugriff zu peripheren Speichern

Es liegt nahe, zunächst nach Verfahren zu suchen, welche die Zugriffszeit zu peripheren Speichern verkürzen. Als Beispiele sollen Trommel- und Plattenspeicher angeführt werden.

3.3.1.1 Zugriff zum Trommelspeicher

Coffman [43] analysiert die Wartezeiten für den Zugriff auf einen Trommelspeicher in einem Paging-Betriebssystem. Die Trommeloberfläche ist längs der Achse in N Sektoren aufgeteilt, welche jeweils soviele Seiten speichern können, wie es Sätze von Schreib-/Leseköpfen (Felder) gibt. Schreib-/Lesewünsche werden von einem E/A-Prozessor in Sektorspezifischen Warteschlangen gespeichert und werden nur dann ausgeführt, wenn S/L-Kopf und Sektor übereinanderliegen, siehe Bild 5. Unter Annahme Poisson-verteilter page-Anfragen wird mit Hilfe einer eingebetteten Markoff'schen Kette die Analyse der Wartezeiten durchgeführt.

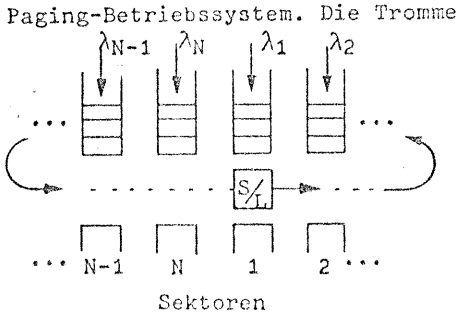

Bild 5. Modell des Zugriffs zum Trommelspeicher

In einer Arbeit von Abate und Dubner [44] wird der Trommelzugriff optimiert, indem durch Einführung eines sog.

"hardware queuers" die Anfragen nicht in der Reihenfolge des Eintreffens, sondern in jener Reihenfolge bearbeitet werden, welche unter Berücksichtigung der Drehrichtung eine minimale Zugriffszeit zur Folge haben. Das System des "hardware queuers" wird analytisch näherungsweise behandelt.

3.3.1.2 Zugriff zum Plattenspeicher

Frank [45] untersucht Plattenspeicherzugriffe bei Speichern mit einem und mehreren Schreib-/Leseköpfen je Oberfläche. Um die mittlere Zugriffszeit zu verringern, wird eine Strategie vorgeschlagen, wie einzelne Datenblöcke (records) entsprechend ihrer Zugriffshäufigkeit auf der Platte optimal gespeichert bzw. von der Platte optimal gelesen werden müssen. Als Maß für die Leistungsfähigkeit des Systems werden Erwartungswerte für die Einstellzeit des Arms auf die Spur und die Rotationsverzögerung angegeben. In einer Arbeit von Abate et al [46] wird ein Plattenspeichersystem mit Hilfe eines zweistufigen (Tandem-) Warteschlangenmodells untersucht. Unter einigen vereinfachenden Voraussetzungen wird die Verteilungsfunktion der Antwortzeit des Systems angegeben. Die Antwortzeit setzt sich hierbei zusammen aus der Wartezeit für den Arm, der Einstellzeit des Arms auf die Spur (1. Bedienungseinheit), der Wartezeit auf den Übertragungskanal sowie der Übertragungszeit des Kanals (2. Bedienungseinheit).

3.3.2 Modelle für die Speicherplatzverwaltung innerhalb von Speichern

Bei der Speicherplatzverwaltung (z.B. eines Realzeitsystems zur Meßwertverarbeitung) müssen einzelne Worte (Meßwerte, Befehle) bzw. ganze Blöcke ein- und ausgespeichert werden. Hierfür sind verschiedene Methoden gebräuchlich, vergl. z.B. Martin [47]. In Bild 6 sind drei bekannte Strategien zur Speicherplatzbelegung bzw. Speicherplatzlöschung skizziert.

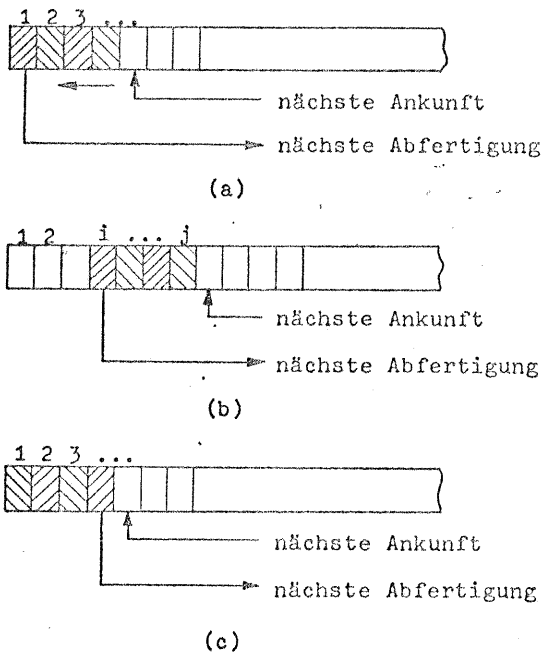

Bild 6. Strategien zur Speicherplatzverwaltung

Bild 6a zeigt zunächst eine normale FIFO-Organisation. Eine neu in den Speicher eintreffende Einheit (Wort, Block) wird am Ende der Warteschlange eingereiht. Es wird stets die Einheit, welche den ersten Platz belegt, abgefertigt. Verläßt eine wartende Einheit den Speicher, so rücken alle Wartenden um einen Platz nach vorn. In Bild 6b ist eine modifizierte FIFO-Organisation gezeigt, welche das Umspeichern nach jeder Abfertigung vermeidet. Der gesamte belegte Speicherplatz bildet - wie bei Bild 6a - eine fortlaufende Folge von Adressen, er wandert jedoch im Verlauf der Zeit in dem linear gedachten Speicherbereich nach rechts. Bei jedem zufälligen Ereignis "Speicher leer" beginnt der Prozess wieder von vorn bei Adresse 1. In Bild 6c ist schließlich die LIFO-Organisation dargestellt, bei welcher eine eintreffende Einheit an den Kopf der Warteschlange gesetzt wird; die Abfertigung erfolgt ebenfalls vom Kopf der Warteschlange. Auf diese Weise bleibt stets ein Ende der Warteschlange fest, es braucht keine Umspeicherung vorgenommen zu werden.

Die drei dargestellten Strategien unterscheiden sich hinsichtlich des Organisationsaufwands (Suchen von Anfangsadressen, Verschieben nichtleerer Warteschlangen) wie auch in Bezug auf den Speicheraufwand. Bei begrenztem Speicherplatz sind insbesondere die Strategien Bild 6a und 6c günstig.

Coffman und McKellar [48] geben Lösungen für Verteilungen der Warteschlangenlänge sowie der Position von Kopf und Schwanz der Schlange für das Modell Bild 6b an. Unter der Voraussetzung, daß Ankünfte bzw. Abfertigungen zu taktsynchronen Zeitpunkten mit gewissen Wahrscheinlichkeiten auftreten, wird für den Lösungsweg das Verfahren der eingebetteten Markoff-Kette angewendet. Das Modell Bild 6a wurde von Coffman und Schmookler [49] mit Hilfe eines Random-Walk Modells behandelt.

Kühn [50,27] untersucht das Modell Bild 6c im allgemeineren Falle begrenzten Speicherraums für verdrängende und nichtverdrängende Prioritäten unter rein Markoff'schen Voraussetzungen. Erweiterungen der Modelle Bild 6a und 6c auf mehrere Warteschlangen mit begrenztem Speicherraum wurden ebenfalls von Kühn [4,27] vorgenommen und analytisch behandelt.

Es sei an dieser Stelle ferner auf Warteschlangenmodelle hingewiesen, welche Aufschlüsse über erforderliche Speichergrößen für jene Fälle geben, bei denen die Bedienungseinheit die eingespeicherten Einheiten in konstanten Zeitabständen abholt. Dor [51] untersucht die erforderliche Größe eines Pufferspeichers unter der Annahme Poisson-verteilter Ankünfte. Langenbach-Belz [52] behandelt das Problem von g >1 zyklisch in konstanten Zeitabständen abgefragten Speichern. Beide Untersuchungen wenden das Verfahren der eingebetteten Markoff-Kette an.

3.3.3 Modelle für den Verkehr innerhalb einer Speicherhierarchie

Moderne Rechnersysteme für Multiprogramming- und Time-Sharing-Betrieb verwenden ausnahmslos eine den Zugriffszeiten entsprechend abgestufte Speicherhierarchie (z.B. Schnellspeicher - Arbeitsspeicher - Trommelspeicher). Durch eine zweckmäßige Organisation zur Überlappung von Prozessor- und E/A-Aktivitäten (Multiprogramming) ist es möglich, dem zentralen Prozessor einen sehr großen virtuellen Arbeitsspeicher zu geben. Auf den Verkehr zwischen Prozessor(en) und der Speicherhierarchie wird in Abschnitt 3.4 noch ausführlicher eingegangen werden; hier sollen für den Verkehr zwischen verschiedenen Ebenen innerhalb der Speicherhierarchie einige Verkehrsmodelle aufgezeigt werden.

3.3.3.1 Blocktransfer zwischen peripheren Speichern und Arbeitsspeicher

Voraussetzung für einen hohen Grad an Simultanarbeit ist ein in weitgehend autonome Moduln aufgeteilter Arbeitsspeicher. Hierdurch ist es im Multiprogramming-Betrieb möglich, simultan Arbeitsspeichermoduln zu laden und Programme anderer Arbeitsspeichermoduln zu bearbeiten.

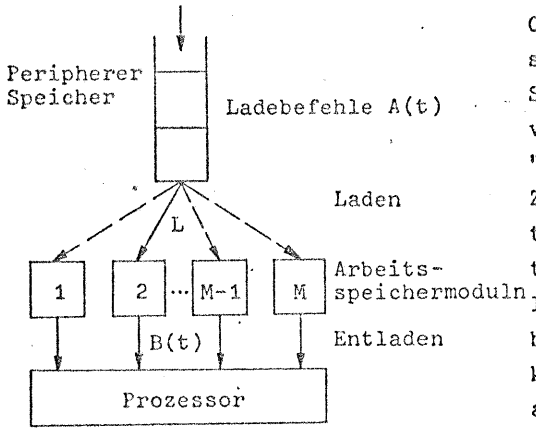

Bild 7. Laden von Arbeitsspeichermoduln durch peripheren Speicher

Coffman gibt folgendes Modell Bild 7 an. Der Arbeitsspeicher besteht aus M Moduln, welche z.B. jeweils eine Seite (page) aufnehmen können. Jeder Modul kann einen von drei Zuständen annehmen: "ladend", "verfügbar" und "aktiv". Es kann jeweils nur ein Modul zur gleichen Zeit geladen werden, während mehrere Moduln gleichzeitig aktiv sein können, d.h. durch die Programmbearbeitung "entladen" werden können. Ladebefehle sowie Entladezeiten können durch allgemeine Verteilungen A(t) bzw. B(t) beschrieben werden. Die Ladezeit L kann als konstant angenommen werden. Coffman [53] bestimmt auf analytische Weise die obere Durchsatzgrenze unter Voraussetzung der Sättigung des Hintergrunds mit wartenden Programmseiten und negativ-exponentiell verteilter Entladezeiten für jeden aktiven Modul.

Der Blocktransfer zwischen peripherem Speicher und Arbeitsspeicher wirft u.a. eine Reihe von interessanten Fragen auf:

- Größe der Seiten
  Kleine Seiten bedingen höheren Verwaltungsaufwand, dafür aber weniger "Speicherverschnitt".
- Anzahl gleichzeitig anwesender Seiten im Arbeitsspeicher
  Je mehr Benutzerprogramme simultan bearbeitet werden, desto weniger Seiten je Programm stehen im Arbeitsspeicher. Dadurch häufen sich die Fälle, in denen Programmteile oder Daten benötigt werden ("page-faults"), welche auf peripheren Speichern liegen und eine E/A-Aktion , und damit Systemzeit, beanspruchen (swapping).
- Strategie des Seitentransfers
  Die "Lokalitäts-Eigenschaft" der meisten Benutzerprogramme läßt es günstig erscheinen, im Falle eines page-faults gleich mehrere Seiten auszutauschen. Eine weitere Frage ist die Strategie, nach welcher die Seiten ausgetauscht werden sollen (z.B. "least-recently-used").

Kuck und Lawrie [54] untersuchen ausführlich die Auswirkungen von Parametern und Strategien auf Rechnersysteme mit Paging-Betriebssystemen. Shemer und Shippey [55] stellen statistische Modelle für den Seitentransfer auf, welche mit Hilfe von Simulationen ausgewertet werden. Speziell wer-

den hierbei ein Arbeitsspeicher sowie ein Schnellspeicher analysiert (Ausführliche Literaturangaben zu diesem Thema findet man in [54]). Über die vielfältigen Adressierungsprobleme wird von Swoboda [72] berichtet.

3.3.3.2 Puffermethoden

Zur schnelleren Eingabe von Programmen und Daten in den Arbeitsspeicher sind verschiedene Methoden vorgeschlagen worden, welche einen hohen Grad an Simultanarbeit ermöglichen.

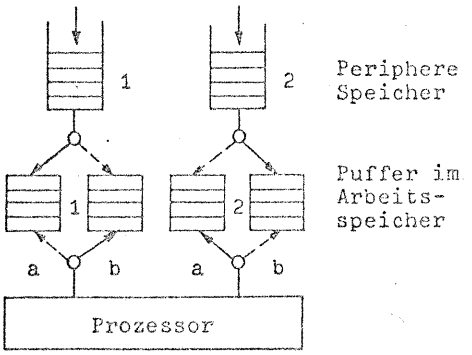

Bild 8. Wechselpuffermethode

Bild 8 zeigt die Methode der Wechselpuffer (double buffering). Jedem peripheren Speicher ist ein Doppelpufferbereich im Arbeitsspeicher reserviert. Während z.B. Puffer 1a geladen wird, kann Puffer 1b entladen werden und umgekehrt. Einen analytischen Ansatz zur Abschätzung der Leistungsfähigkeit der Wechselpuffermethode gibt Gaver [56].

In einer Arbeit von Woodrum [57] wird in Erweiterung zur Wechselpuffermethode ein Konzept mit "fließenden Puffern" vorgeschlagen. Die Analyse erfolgt mit Hilfe eines Semi-Markoff-Prozesses.

3.4 Modelle für den Verkehr zwischen verschiedenartigen Funktionseinheiten

Zur Untersuchung der Leistungsfähigkeit eines Rechnersystems ist es neben der Untersuchung von Teilsystemen, wie sie in den Abschnitten 3.2 und 3.3 behandelt wurden, unbedingt erforderlich, Gesamtmodelle für das Zusammenspiel verschiedenartiger Funktionseinheiten aufzustellen. Es gibt eine Reihe von Ansätzen, das statistische Verkehrsverhalten dieser G e s a m tmodelle auf analytische Weise zu beschreiben. Einen Überblick über Modellkriterien und analytische Methoden findet man beispielsweise bei Chang [58]. Häufig wird jedoch zur Untersuchung der Leistungsfähigkeit von Rechnersystemen als Ganzes die Simulation angewendet. Kümmerle [17] stellt hierfür alle wichtigen Kriterien zusammenfassend gegenüber. (Für einschlägige Literaturangaben über Analyse- und Simulationsverfahren sei ebenfalls auf [58] und [17] verwiesen).

Die Modelle für den Verkehr zwischen verschiedenartigen Funktionseinheiten seien in zwei Hauptgruppen eingeteilt:

- Modelle für den Verkehr zwischen Prozessor(en) und Peripherie
- Modelle für den Verkehr zwischen Prozessor(en) und Arbeitsspeichermoduln.

(Die Abgrenzung zwischen den verschiedenen Hauptgruppen bzw. zwischen diesen Hauptgruppen und vorausgegangener Modelle läßt sich nicht immer einheitlich durchführen).

3.4.1 Modelle für den Verkehr zwischen Prozessor(en) und Peripherie

Der Multiprogramming- und Time-Sharing-Betrieb erfordert einen umfangreichen Verkehr zwischen verschiedenartigen Funktionseinheiten innerhalb eines Rechnersystems. Leipold und Spreen [59] klassifizieren allgemein den Nachrichtenverkehr zwischen den Funktionseinheiten "Zentralprozessor", "Arbeitsspeicher", "E/A-Kanal" sowie "Peripheriegerät" unter Berücksichtigung der speziellen Schnittstellenbedingungen. Über die Organisation des Multiprogramming-Betriebs mit paging sei auf eine Arbeit von Randell und Kuehner [60] verwiesen. Das Ziel der Analyse von Verkehrsmodellen für das Ablaufgeschehen in Rechnersystemen ist es, unter Berücksichtigung wirklichkeitsnaher "Verkehre" Strategien zu finden, welche die einzelnen Funktionseinheiten gut ausnutzen und einen wirtschaftlichen Kompromiss zwischen Verweilzeit einer Anfrage im System und Durchsatz darstellen.

In Bild 9 ist allgemein die Systemkonfiguration beim Paging-Verfahren dargestellt. Die Speicherung erfolgt in 2 Ebenen: die peripheren Speicher enthalten alle Programmsegmente der Anwenderprogramme. Im Arbeitsspeicher ist, neben dem residenten Teil des Betriebssystems BS, nur eine beschränkte Zahl N von Seiten gespeichert. Der zentrale Prozessor bearbeitet die i.a. zu mehreren (konkurrierenden) Anwenderprogrammen gehörigen Seiten im Arbeitsspeicher. Kann eine Information nicht im Arbeitsspeicher gefunden werden (page-fault), so wird ein Seitentransfer veranlaßt, welcher eine oder mehrere Seiten des betreffenden Programmes in den Arbeitsspeicher

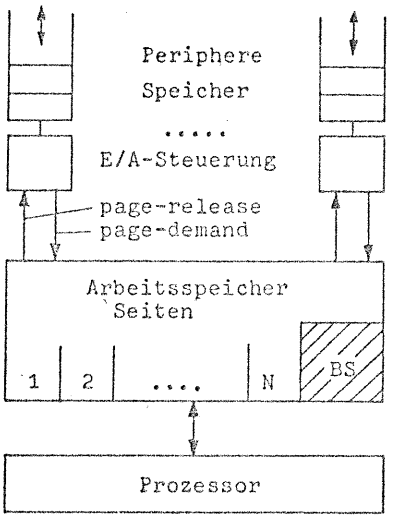

Bild 9. Systemkonfiguration beim Paging-Verfahren

holt (demand paging). Das Betriebssystem hat nach einer vorgegebenen Strategie zu entscheiden, w e l c h e "alten" Seiten im Arbeitsspeicher überschrieben werden (replacement strategy). Sobald ein Seitentransfer notwendig wird, wendet sich der Prozessor einem anderen Anwenderprogramm zu.

Anacker und Wang [61] charakterisieren den Verkehrsfluß zwischen den Funktionseinheiten unter Berücksichtigung verschiedener Systemkonfigurationen und Anwenderprogramme durch Untersuchung der Adresspfade von ausgeführten Programmen. Es wird eine höchstmögliche Verarbeitungsrate angegeben.

In einer Analyse von Gaver [56] werden Abhängigkeiten zwischen Prozessorgeschwindigkeit, Speichergröße sowie Seitenzahl und Zahl der E/A-Einheiten analytisch untersucht. Als Ergebnis wird der Erwartungswert der Prozessorausnutzung für verschiedene Verteilungen der Programmtypen und Programmlängen angegeben. Die Analyse erfolgt für ein Modell, welches vier mögliche Zustände einer Seite betrachtet: "In Bearbeitung"; "Warten auf Prozessor", "E/A-Aktivität" und "Warten auf E/A-Kanal". Es wird die Auslastung des Prozessors, d.h. der Erwartungswert der Betriebs-(Blockierungs-)periode berechnet.

In einer Arbeit von Shedler und Yang [62] wird ein Simulationsmodell entworfen, welches zusätzlich eine Reihe von Betriebssystemoperationen (system overhead) berücksichtigt (z.B. Umschaltzeiten des Prozessors auf ein anderes Programm, Ausführen der Seiten-Ersetzung u.a.m.). Das Modell ist ein zweistufiges rückgekoppeltes Wartesystem, wobei sich die "Bedienung" in jeder Stufe aus drei Phasen zusammensetzt: der eigentlichen Prozessorzeit, ferner der Seitentransferzeit, jede zuzüglich zweier Betriebssystem-Phasen.

Über Fragen der Systemkonfiguration und des Einsatzes eines externen Kernspeichers für ein Multiprogramming-Betriebssystem berichten Campbell et al [63]. Die Analyse der Leistungsfähigkeit wurde hierbei nach der Methode von Gaver [56] durchgeführt.

Courtois und Georges [64] schlagen ein Modell entsprechend Bild 10 für den Multiprogramming-Betrieb vor. Der Prozessor kann nur mit dem Primärspeicher M_0 direkt in Verbindung treten, während die Informationen der nach Zugriffszeiten abgestuften Sekundärspeichertypen $M_1,\ldots,M_L$ erst über die Kanäle $C_1,\ldots,C_L$ in den Primärspeicher M_0 gebracht werden müssen. Umgekehrt müssen von M_0 "suspendierte" Programme über diese Kanäle in einen der Sekundärspeicher umgespeichert werden. Diese Umspeicherung erfolgt nach der Strategie, daß die Information mit dem am längsten zurückliegenden Zugriff in den langsamsten Sekundärspeicher abgelegt wird (least-recently-used-strategy). Unter Annahme negativ-exponentiell verteilter Übertragungszeiten wird eine durch fortschreitende Betrachtung von Teilsystemen gewonnene Lösung angegeben.

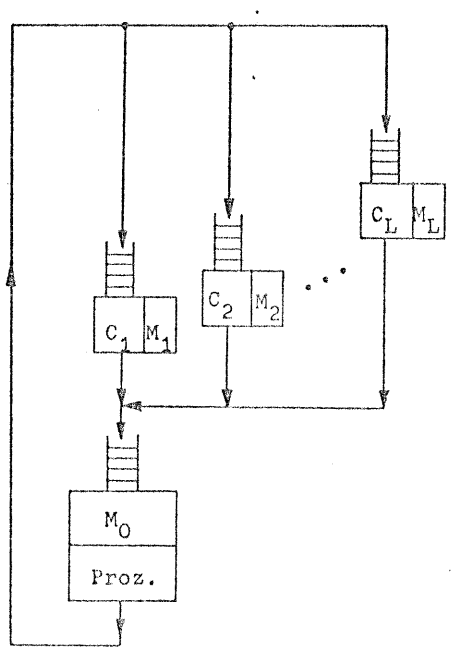

Bild 10. Multiprogramming-Modell mit abgestufter Speicherhierarchie

In einem Beitrag von Spiess [65] wird ein Modell zur Simultanarbeit von einem Prozessor und einem E/A-Kanal analytisch untersucht. Das Modell Bild 11 besteht aus den Bedienungseinheiten "Prozessor" und "E/A-Kanal". Jobs ordnen sich zunächst in die Prozessor-Warteschlange ein. Nach einer Prozessor-Bedienungsphase ist der Job mit Wahrscheinlichkeit p_1 fertig (ohne Berücksichtigung einer Phase für das Ausgeben des Programms und der Ergebnisse), mit Wahrscheinlichkeit $1-p_1$ schließt sich eine E/A-Phase (z.B. Zugriff auf langsamen Hintergrund-

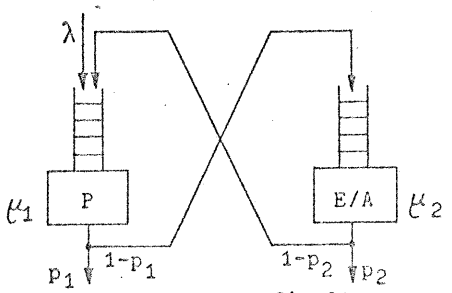

Bild 11. Modell zur Simultanarbeit von Prozessor und E/A-Kanal

speicher) an. Nach einer E/A-Phase ist der Job mit Wahrscheinlichkeit p_2 fertig, mit Wahrscheinlichkeit $1-p_2$ schließt sich eine weitere Prozessorphase an usw. Unter Annahme rein Markoff'scher Voraussetzungen werden Lösungen für Zustandswahrscheinlichkeiten und mittlere Warteschlangenlängen angegeben.

In einer weiteren Arbeit von Spiess [66] wird ein Modell mit einem Prozessor, einer E/A-Einheit sowie einer dreistufigen Speicherhierarchie für den Time-Sharing-Betrieb vorgeschlagen. Es werden Simulationsergebnisse für verschiedene Strategien angegeben.

3.4.2 Modelle für den Verkehr zwischen Prozessor(en) und Arbeitsspeichermoduln

In Rechnersystemen mit mehreren Prozessoren oder/und mit Betriebsarten zur vorausschauenden ("look-ahead") Versorgung des Rechnerkerns mit Befehlen und Operanden wird der Arbeitsspeicher stets in M Moduln unterteilt. Damit ist u.a. möglich, mehrere Zugriffswünsche zeitlich überlappt ablaufen zu lassen, wenn aufeinanderfolgende Adressen bzw. Blocks von Adressen in aufeinanderfolgenden Moduln zyklisch abgelegt sind (Adressenverschränkung). Der Verkehr zwischen den Prozessoren und Speichermoduln erfolgt i.a. über ein Verbindungsnetzwerk (Bussystem aus einer oder mehreren Busleitungen).

Swoboda [67] diskutiert für ein Multiprozessorsystem mit verschränkten Arbeitsspeichermoduln und zusätzlichen E/A-Werken (vergl. Bild 12) verschiedene Lösungen für das Verbindungsnetzwerk, welche dazu geeignet sind, die Verkehrshemmungen zu erniedrigen. In einer Arbeit von Hofmann [68] wird ein derartiges System analytisch wie auch mit Hilfe der Simulation untersucht. Dabei wird angenommen, daß die Prozessoren (Verkehrsquellen) unabhängig voneinander Zugriffswünsche auf die Arbeitsspeichermoduln erzeugen. Die Modulbelegungszeit wurde konstant und als ganzzahliges Vielfaches der Bus-Belegungszeit vorausgesetzt. Als Ergebnisse werden u.a. Abhängigkeiten des Wirkungsgrades (Verhältnis von Durchsatz zu Gesamtangebot) von Systemparametern (Quellen-, Modul- und Leitungszahl) angegeben.

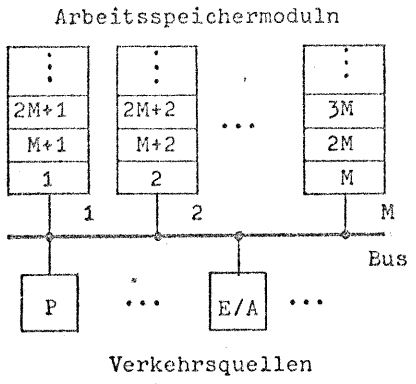

Bild 12. Multiprozessorsystem mit verschränktem Arbeitsspeicher

Eine ausführliche Beschreibung einer komplexen Systemstruktur zur überlappten Befehls- und Operandenversorgung des Prozessors ("pipelining") in einem System mit umfangreicher Speicherhierarchie wurde (einschließlich Angabe von Simulationsergebnissen) von Boland et al [69] veröffentlicht. Über erste Ansätze zur analytischen Behandlung von Look-Ahead-Mechanismen wurde in einer Arbeit von Flores [70] berichtet. Swoboda [71] untersucht die Auswirkungen des "look-ahead" auf die Leistungsfähigkeit eines Rechnersystems mit und ohne Schnellspeicher.

Shemer und Gupta [73] untersuchen die Leistungsfähigkeit eines Prozessors mit Look-Ahead-Organisation, wobei außer vom Prozessor noch gleichzeitig von einer E/A-Steuerung Zugriffswünsche auf einzelne Arbeitsspeichermoduln erzeugt werden. In Bild 13 ist die Systemkonfiguration schematisch dargestellt.

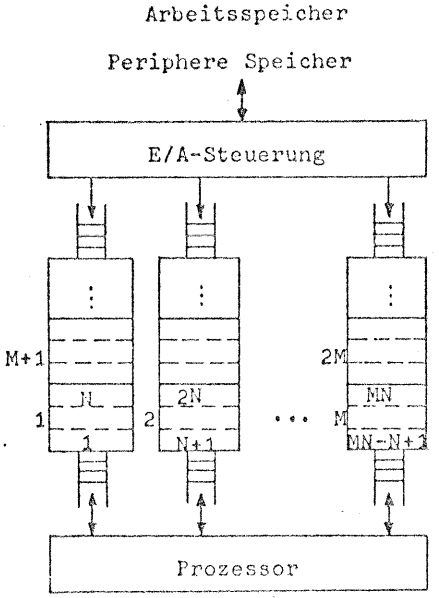

Bild 13. Zugriff von Prozessor und E/A-Steuerung auf adressenverschränkte Arbeitsspeichermoduln

In M Speichermoduln sind aufeinanderfolgende Blöcke der Länge N adressenverschränkt abgelegt. Während einer Befehls-Holphase des Prozessors wird in einem Speicherzyklus ein ganzer Block, bestehend aus N Adressen ("look-ahead"), gelesen. Hiernach schließen sich N einzelne Phasen für das Holen bzw. Abspeichern von Operanden an. Es können hierbei mehrere

Arbeitsspeichermoduln gleichzeitig auf Befehle des Prozessors reagieren. Zugriffswünsche der E/A-Steuerung auf die einzelnen Arbeitsspeichermoduln werden in entsprechende Warteschlangen eingereiht und nach einem FIFO-Modus abgefertigt; sie besitzen nichtunterbrechende Priorität gegenüber Prozessor-Zugriffswünschen.

Die Analyse des Systems wird unter Annahme negativ-exponentiell verteilter Anfrageabstände der E/A-Steuerung sowie einigen vereinfachenden Voraussetzungen näherungsweise durchgeführt. Als Ergebnisse werden mittlere Wartezeiten für Prozessorzugriffe auf Speichermoduln sowie ein Maß für die Zunahme des Prozessor-Durchsatzes im Vergleich zu Systemen ohne unterteilten Arbeitsspeicher angegeben.

4. ANALYSE EINES REALZEIT-PROZESSORMODELLS MIT ZEITABHÄNGIGER UNTERBRECHUNGSSTRATEGIE

4.1 Problemstellung und Voraussetzungen

Es wird ein Realzeit-Rechnersystem mit zwei Arten von Aufgaben betrachtet: Realzeit- oder Vordergrundaufgaben sowie bezüglich der Antwortzeit weniger wichtige Hintergrundaufgaben. Um eine vorgeschriebene Antwortzeit für die Realzeitaufgaben einzuhalten, erhalten diese unterbrechende Priorität. Die Unterbrechungsstrategie sei zeitabhängig (preemption delay): Eine eintreffende Realzeitanfrage unterbricht eine Hintergrundaufgabe nach der konstanten Zeit d, wenn diese nicht schon vorher beendet ist. Die preemption delay-Strategie ermöglicht (bei Einhaltung der vorgeschriebenen Antwortzeit) eine optimal geringe Zahl von Unterbrechungen.

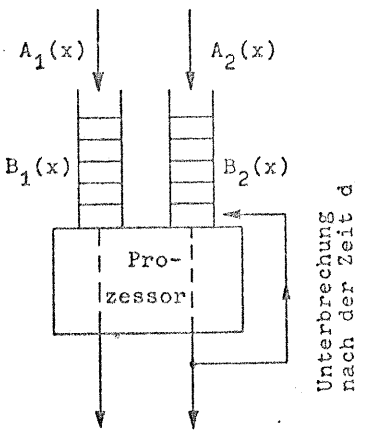

Bild 14. Prozessormodell unter preemption delay-Strategie
Index 1: Realzeit-Aufg.
Index 2: Hintergr.-Aufg.

Bild 14 zeigt das zu analysierende Prozessor-Modell. Es werden folgende Annahmen bezüglich der Anrufabstands- und Bedienungszeitverteilungen getroffen (Definition vergl. Kap. 2):

$$P\{a_i \leq x\} = A_i(x) = 1 - \exp(-\lambda_i x)\ ,\ i = 1,2. \tag{4.1}$$

$$P\{b_1 \leq x\} = B_1(x) \quad \text{beliebig} \qquad E(b_1) = \frac{1}{\mu_1}. \tag{4.2}$$

$$P\{b_2 \leq x\} = B_2(x) = 1 - \exp(-\mu_2 x)\ ,\ E(b_2) = \frac{1}{\mu_2}. \tag{4.3}$$

Die Unterbrechungsstrategie läßt sich durch die Verzögerung v, welche eine Realzeitanfrage durch eine in Bearbeitung stehende Hintergrundaufgabe erleidet, beschreiben:

$$P\{v \leq x\} = V(x) = \begin{cases} 1 - \exp(-\mu_2 x), & 0 \leq x < d \\ 1 & x \geq d. \end{cases} \tag{4.4}$$

Der Parameter d kann beliebig variiert werden; für d = 0 erhält man die rein unterbrechende (preemption) Strategie, für $d \to \infty$ die rein nichtunterbrechende (nonpreemption oder head of the line) Strategie. Es wird ferner vorausgesetzt, daß der Warteraum beliebig groß ist. Die Abfertigung innerhalb der Warteschlangen erfolge nach der Disziplin FIFO. Die Teilangebote $\varrho_i = \lambda_i/\mu_i$, i = 1,2, unterliegen im stationären Falle der Einschränkung $\varrho = \varrho_1 + \varrho_2 < 1$.

4.2 Analyse der Wartezeiten von Realzeitaufgaben

Die Analyse der Realzeitaufgaben erfolgt mit Hilfe der Methode der eingebetteten Markoff-Kette. Die Regenerationspunkte der Kette sind die Zeitpunkte unmittelbar nach Ausscheiden einer Realzeitaufgabe. Der Zustand k beschreibt die Zahl der Realzeitaufgaben im System an den Regenerationspunkten.

Zur Beschreibung wird für jene Realzeitanfrage, welche jeweils den zeitabhängigen Unterbrechungsmechanismus auslöst ("erste Realzeitanfrage"), eine scheinbare Bedienungszeit $f = v + b_1$ eingeführt mit der Verteilungsfunktion

$$P\{f \leq x\} = F(x) = [1 - P_V(d)] \cdot B_1(x) + P_V(d) \cdot V(x) * B_1(x). \tag{4.5}$$

In Gl.(4.5) bedeutet $P_V(d)$ die Wahrscheinlichkeit, daß eine erste Realzeitanfrage auf eine Hintergrundbelegung des Prozessors stößt.

Die Wahrscheinlichkeiten p_{ik} für den Übergang des Zustands i in den Zustand k sind

$$p_{0k} = \int_0^\infty \frac{(\lambda_1 x)^k}{k!} \cdot \exp(-\lambda_1 x) dF(x), \qquad k \geq 0, \tag{4.6a}$$

$$p_{ik} = \int_0^\infty \frac{(\lambda_1 x)^{k-i+1}}{(k-i+1)!} \cdot \exp(-\lambda_1 x) dB_1(x), \qquad i \geq 1,\ k \geq i-1. \tag{4.6b}$$

Die erzeugende Funktion G(s) der Zustandswahrscheinlichkeiten P_i

$$G(s) = \sum_{i=0}^{\infty} P_i \cdot s^i, \quad |s| \leq 1, \tag{4.7a}$$

ergibt sich zu

$$G(s) = P_0 \cdot \Psi_1(\lambda_1 - \lambda_1 s) \cdot \frac{P_V(d) \cdot s \cdot \phi(\lambda_1 - \lambda_1 s) + [1 - P_V(d)] \cdot s - 1}{s - \Psi_1(\lambda_1 - \lambda_1 s)}, \tag{4.7b}$$

wobei $\Psi_1(s)$ bzw. $\phi(s)$ die Laplace-Stieltjes-Transformierten von $B_1(x)$ bzw. V(x) sind. Mit G(1) = 1 folgt zunächst aus Gl.(4.7b)

$$P_0 = \frac{1 - \rho_1}{1 + P_V(d)\lambda_1 \cdot E(v)}. \tag{4.8}$$

Mit $G'(1) = \lambda_1 \cdot E(r_1)$ und $E(w_1) = E(r_1) - E(b_1)$, wobei w_1 die Wartezeit, r_1 die Antwortzeit für Realzeitanfragen bedeuten, folgt:

$$E(w_1) = t_{w_1} = \frac{\lambda_1}{2} \cdot \frac{E(b_1^2)}{1 - \rho_1} + P_V(d) \cdot \frac{E(v) + \frac{1}{2}\lambda_1 E(v^2)}{1 + P_V(d)\lambda_1 \cdot E(v)}. \tag{4.9}$$

Mit Hilfe der erzeugenden Funktion können noch weitere Aussagen über die Verteilung der Wartezeiten bzw. deren höhere Momente gemacht werden.

4.3 Analyse der Wartezeiten von Hintergrundaufgaben

Eine ausführliche Analyse kann mit einer zweidimensionalen Markoff-Kette erfolgen. Die mittlere Wartezeit für Hintergrundaufgaben $E(w_2) = t_{w_2}$ kann jedoch bereits mit Hilfe der Momentenmethode bestimmt werden, vergl.2.5.2, indem die Erwartungswerte der Dauern aufeinanderfolgender Phasen aufsummiert werden. Man erhält auf diese Weise

$$E(w_2) = t_{w_2} = \frac{1}{1 - \rho_1 - \rho_2} \cdot \left[\rho_1 \cdot P_V(d) \cdot \frac{E(v) + \frac{1}{2}\lambda_1 E(v^2)}{1 + P_V(d)\lambda_1 \cdot E(v)} + \frac{1}{2}\lambda_1 \cdot \frac{E(b_1^2)}{1 - \rho_1} + \frac{1}{2}\lambda_2 \cdot E(b_2^2) + \frac{\rho_1}{\mu_2} \exp(-\mu_2 d) \right]. \tag{4.10}$$

4.4 Verzögerungs-(Delay-) Wahrscheinlichkeit

Die in den Ergebnissen für t_{w_1} und t_{w_2} enthaltene Wahrscheinlichkeit $P_V(d)$ läßt sich aus der Überlegung bestimmen, daß eine eintreffende Realzeitanfrage mit Wahrscheinlichkeit $P_0 \cdot P_V(d) = P_0 - (1 - \rho_1 - \rho_2)$ verzögert wird. Hieraus folgt mit Gl.(4.8) und (4.4):

$$P_V(d) = \frac{\rho_2}{1 - \rho_1 + \frac{\lambda_1}{\mu_2}(1 - \rho_1 - \rho_2)[1 - \exp(-\mu_2 d)]}. \tag{4.11}$$

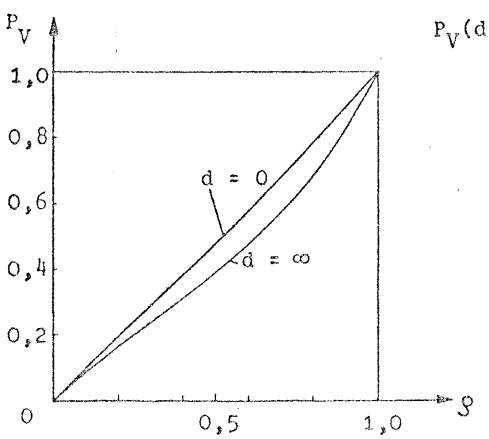

Bild 15. Abhängigkeit der Delay-Wahrscheinlichkeit von d, ρ

4.5 Diskussion der Ergebnisse

Es werde ein Beispiel mit folgender Parameterkombination betrachtet, welches für Realzeitsysteme mit Hintergrundaufgaben typisch ist: $\rho_2 = 10\rho_1$, $\lambda_1 = \lambda_2$, $1/\mu_1 = 1$ Zeiteinheit, $1/\mu_2 = 10$ Zeiteinheiten, $B_1(x)$ hyperexponentiell verteilt mit $E(b_1^2) = 3/\mu_1^2$.

In Bild 15 sind zunächst die Grenzwerte für $P_V(d)$ in Abhängigkeit des Angebots ρ angegeben. Man erkennt, daß die Abhängigkeit der Verzögerungs-Wahrscheinlichkeit von d nur gering ist.

Zur Beurteilung der Leistungsfähigkeit der zeitabhängigen Unterbrechungsstrategie ist es zweckmäßig, die Antwortzeiten zu betrachten. In Bild 16 sind Ergebnisse für die mittlere Antwortzeit der Realzeitanfragen $E(r_1) = t_{R_1}$ in Abhängigkeit der Verzögerung d als auch dem Angebot ϱ angegeben.

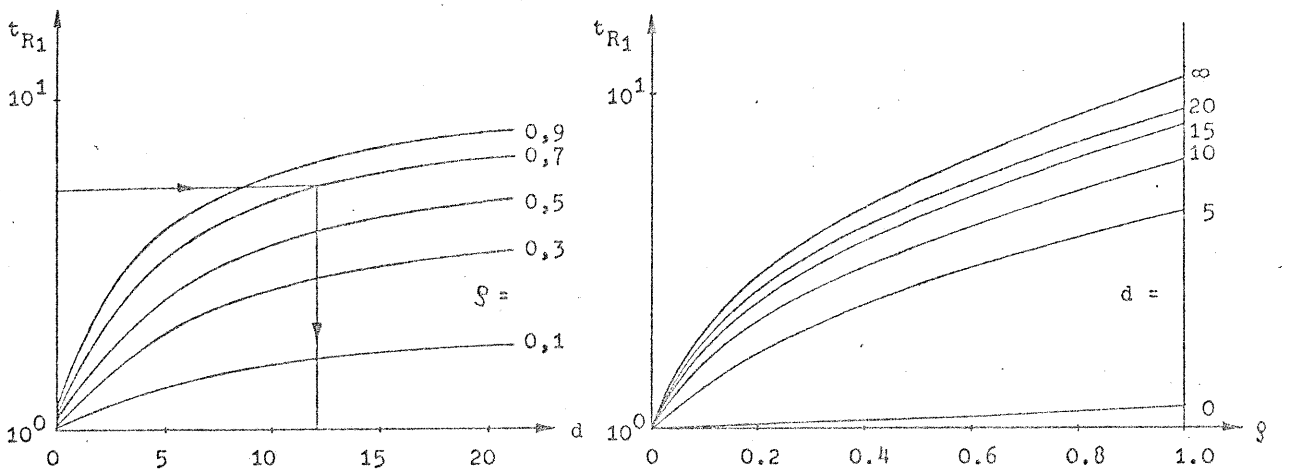

Bild 16. Mittlere Antwortzeit der Realzeitanfragen in Abhängigkeit von d und ϱ

4.6 Optimierung der Zahl der Unterbrechungen

Die mittlere Zahl von Unterbrechungen, welche eine Hintergrundaufgabe während ihrer Aufenthaltsdauer im System erleidet, berechnet sich zu

$$U_m = \frac{\lambda_1}{\mu_2} \cdot \exp(-\mu_2 d). \qquad (4.12)$$

Bei v o r g e s c h r i e b e n e r maximal zulässiger mittlerer Antwortzeit t_{R_1} der Realzeitanfragen und bekanntem Angebot kann mit Hilfe von Bild 16 die optimale Verzögerung d bestimmt werden, vergl. Eintrag in Bild 16. Gegenüber rein unterbrechender Priorität (d = 0) werden mit dieser Strategie im Mittel

$$\Delta U_m = \frac{\lambda_1}{\mu_2} \cdot [1 - \exp(-\mu_2 d)] \qquad (4.13)$$

Unterbrechungen eingespart.

ZUSAMMENFASSUNG

In dem vorliegenden Beitrag wurde der Versuch unternommen, die vielfältigen Modelle zur analytischen Behandlung des Ablaufgeschehens in Rechnersystemen zu klassifizieren. Es hat sich hierbei gezeigt, daß die verschiedenartigen Abläufe auf wenige, relativ standardisierte Warteschlangenmodelle zurückgeführt werden können. Mit Hilfe derartiger Modellvorstellungen ist es möglich, die Leistungsfähigkeit verschiedenartiger Strukturen und Strategien zu vergleichen. Im abschließenden Kapitel wurde für ein zusätzlich mit Hintergrundaufgaben belastetes Realzeitsystem mit zeitabhängiger Unterbrechungsstrategie eine analytische Lösung angegeben.

SCHRIFTTUM

[1] Ferschl,F.: Zufallsabhängige Wirtschaftsprozesse. Physica-Verlag. Wien und Würzburg, 1964.

[2] Syski,R.: Introduction to Congestion Theory in Telephone Systems. Oliver and Boyd. Edinburgh, 1960.

[3] Marte,G.: Optimal Time Scheduling for Time-Shared Computer Systems with Piecewise Exponential Computing Time Distribution. Proc. Intern.Comp.Symp. Bonn, 21.-22.Mai 1970, S.184-206.

[4] Kühn,P.: Parallel Waiting Queues in Real-Time Computer Systems. Proc. Intern.Comp.Symp. Bonn, 21.-22.Mai 1970, S.143-158, sowie NTZ 23(1970),H.11,S.576-582.

[5] Phipps,T.E.: Machine Repair as a Priority Waiting-Line Problem. Op.Res.Vol.9(1961),S.732-742.

[6] Stephan,F.F.: Two Queues under Preemptive Priority with Poisson Arrival and Service Rates. Op.Res. Vol.6(1958), S.399-418.

[7] Gaver,D.P.: A Waiting Line with Interrupted Service, Including Priorities. J. Roy.Stat.Soc. Series B 24 (1962), S.73-90.

[8] Cobham,A.: Priority Assignment in Waiting Line Problems. Op.Res. Vol.2(1954), S.70-76.

[9] Takács,L.: Priority Queues. Op. Res. Vol.12(1964), S.63-74.

[10] Coffman,E.G.: On the Tradeoff Between Response and Preemption Costs in a Foreground-Background Computer Service Discipline. IEEE Transact.Comp. Vol.C-18(1969), S.942-947.

[11] Herzog,U.: Preemption-Distance Priorities in Real-Time Computer Systems. To be Published in NTZ, 1972.

[12] Gaver,D.P.: On Priority Type Disciplines in Queuing. Proc. of the Symp. on Cong. Theory (1964). The University of North Carolina Press. Chapel Hill, 1965, S.228-252.

[13] Jaiswal,K.: Priority Queues. Academic Press. New York and London, 1968.

[14] Morse,P.M.: Queues, Inventories and Maintainance. Publ.in Op.Res.J.Wiley.New York,Vol.1,1958.

[15] Takács,L.: Introduction to the Theory of Queues. Oxford University Press. New York, 1962.

[16] Saaty,T.L.:Elements of Queuing Theory. McGraw-Hill. New York, 1961.

[17] Kümmerle,K.: Simulation of the Performance of Computer Systems. Elektron. Rechenanl. 12(1970),H.6, S.324-328.

[18] Gordon,G.: System Simulation. Prentice Hall Inc. Englewood Cliffs, N.J., 1969.

[19] Kampe,G.: Simscript. Friedr. Vieweg und Sohn. Braunschweig, 1971.

[20] Herzog,U.und Kühn,P.: Comparison of some Multi-Queue Models with Overflow and Load-Sharing Strategies for Data Transmission and Computer Systems. Proceedings of the Symposium on Computer Communications Networks and Teletraffic. Brooklyn, New York, April 4-6, 1972.

[21] Hansen,P.B.: Analysis of Response Ratio Scheduling. Proc. of IFIP Congr. Ljubljana, Aug. 1971, TA-3, S.150-154.

[22] Küspert,H.-J.und Marte,G.: Optimale Rechenzeit-Zuteilung bei einem Teilnehmer-Rechensystem mit jeweils einer Aufgabe im Arbeitsspeicher.Elektron.Rechenanl. 12(1970),H.3,S.155-162.

[23] Avi-Itzhak,B.,Brosh,I.und Noar,P.: On Discretionary Priority Queuing. Zeitschr.f.angew.Math.u.Mech.(ZAMM) 1964, H.6, S. 235-242.

[24] Schrage,L.: Analysis and Optimization of a Queuing Model of a Real-Time Computer Control System. IEEE Transact. Comp. Vol.C-18(1969), S.997-1003.

[25] Olivier,G.: Optimale Zeitzuteilung für wartende Rechenaufgaben. Elektron.Rechenanl. 9(1967), H.5, S.218-224.

[26] Kühn,P.: Combined Delay and Loss Systems with Several Input Queues, Full and Limited Accessibility. AEÜ 25(1971), H.9/10, S.449-454.

[27] Kühn,P.: Über die Berechnung der Wartezeiten in Vermittlungs- und Rechnersystemen. Diss. Universität Stuttgart 1972.

[28] Kümmerle,K.: Aufbau und Wirkungsweise von Teilnehmer-Rechensystemen. ETZ-B 21(1969), H.22, S.511-515.

[29] Hellerman,H.: Some Principles of Time-Sharing Scheduler Strategies. IBM Syst. Journ. No.2(1969), S.94-117.

[30] Kleinrock,L.: Swap-Time Considerations in Time-Shared Systems. IEEE Transact. Comp. Vol. C-19(1970), S.534-540.

[31] Coffman,E.G.und Kleinrock,L.: Some Feedback Queuing Models for Time-Shared Systems. Proc. of 5th ITC. New York, June 14-20,1967, S.288-304.

[32] Kleinrock,L.und Muntz,R.R.: Multilevel Processor-Sharing Queuing Models for Time-Shared Systems. Proc. of 6th ITC. München, Sept. 9-15,1970, S.341/1-8.

[33] Kleinrock,L.: A Selected Menu of Analytic Results for Time-Shared Computer Systems. IBM Comp. Science Seminar. Stuttgart, Sept. 18, 1970.

[34] Coffman,E.G.,Muntz,R.R.und Trotter,H.: Waiting Time Distributions for Processor-Sharing Systems. J.ACM Vol.17(1970),No.1, S.123-130.

[35] Walke,B.und Küspert,H.-J.: Teilnehmer-Rechensysteme: Mittlere Verweilzeiten bei optimaler Rechenzeitzuteilung. Elektron.Rechenanl.13(1971), H.5, S.193-199.

[36] Chang,W.: Queues with Feedback for Time-Sharing Computer System Analysis. Op. Res. Vol. 15(1967), S.613-627.

[37] Adiri,I.: A Note on Some Mathematical Models of Time-Sharing Systems. J.ACM Vol.18(1971), No.4, S.611-615.

[38] Rasch,P.J.: A Queueing Theory Study of Round-Robin Scheduling of Time-Shared Computer Systems. J.ACM Vol.17(1970), No.1, S.131-145.

[39] Coffman,E.G.: Analysis of Two Time-Sharing Algorithms Designed for Limited Swapping. J.ACM Vol. 15(1968), No.3, S.341-353.

[40] Kleinrock,L.,Muntz,R.R.und Hsu,J.: Tight Bounds on the Average Response Time for Time-Shared Computer Systems.Proc.of IFIP Congr. Ljubljana, Aug.1971, TA-2, S.50-58.

[41] Adiri,I.: A Dynamic Time-Sharing Priority Queue. J.ACM Vol.18(1971), No.4, S.603-610.

[42] Adiri,I.,und Avi-Itzhak,B.: A Time-Sharing Model with Many Queues. Op. Res. Vol. 19(1969), S.1077-1089.

[43] Coffman, E.G.: Analysis of a Drum Input/Output Queue under Scheduled Operation in a Paged Computer System. J. ACM Vol. 16(1969), No. 1, S.73-90.

[44] Abate,J.und Dubner,H.: Optimization of a Drum-Like Storage. IEEE Transact. Comp. Vol. C-18(1969), S.992-997.

[45] Frank,H.: Analysis and Optimization of Disk Storage Devices for Time-Sharing Systems. J. ACM Vol.16(1969), No.4, S.602-620.

[46] Abate,J.,Dubner,H.und Weinberg,S.B.:Queueing Analysis of the IBM 2314 Disk Storage Facility. J. ACM Vol.15(1968), No.4, S.577-589.

[47] Martin,J.:Design of Real-Time Computer Systems.Prentice Hall Inc.Englewood Cliffs,N.J.,1967.

[48] Coffman,E.G. und McKellar,A.C.: On the Motion of an Unbounded, Markov Queue in Random Access Storage. IEEE Transact.Comp. Vol.C-17(1968), S.600-603.

[49] Coffman,E.G. und Schmookler,M.S.: A Random-Walk Model of a Queue Storage Problem. IEEE Transact.Comp. Vol.C-17(1968), S.1093-1095.

[50] Kühn,P.: On a Combined Delay and Loss System with Different Queue Disciplines. Sixth Prague Conf. on Information Theory, Statistical Decision Functions and Random Processes. Prag, 19.-25.Sept.1971 (Proceedings to be published 1972).

[51] Dor,N.M.: Guide to the Length of Buffer Storage Required for Random (Poisson) Input and Constant Output Rates. IEEE Transact.Comp. Vol.C-18(1967), S.683-684.

[52] Langenbach-Belz,M.: Sampled Queuing Systems. Proceedings of the Symposium on Computer Communications Networks and Teletraffic. Brooklyn, New York, April 4-6, 1972.

[53] Coffman, E.G.: A Simple Probability Model Yielding Performance Bounds for Modular Memory Systems. IEEE Transact.Comp. Vol.C-17(1968), S.86-89.

[54] Kuck, D.J. und Lawrie,D.H.: The Use and Performance of Memory Hierarchies: A Survey. Aus: Software Engineering. Computer and Information Sciences, Vol.1. Academic Press. New York/London, 1970, S.45-76.

[55] Shemer,J.E. und Shippey,G.A.: Statistical Analysis of Paged and Segmented Computer Systems. IEEE Transact.Comp. Vol.EC-15(1966), S.855-863.

[56] Gaver,D.P.: Probability Models for Multiprogramming Computer Systems. J. ACM Vol.14(1967), No.3, S.423-438.

[57] Woodrum,L.J.: A Model of Floating Buffering. IBM Syst.Journ.(1971), S.118-144.

[58] Chang,W.: Single-Server Queuing Processes in Computing Systems. IBM Syst.Journ.(1970), S.36-71.

[59] Leipold,K. und Spreen,H.: Organisation des Nachrichtenverkehrs zwischen Zentraleinheiten und peripheren Einheiten in Datenverarbeitungssystemen. Elektron. Rechenanl. 11(1969), H.3, S.151-161.

[60] Randell,B. und Kuehner,C.J.: Dynamic Storage Allocation Systems. Comm. ACM Vol.11(1968), No.5, S.297-306.

[61] Anacker,W. und Wang,C.P.: Performance Evaluation of Computing Systems with Memory Hierarchies. IEEE Transact.Comp. Vol.EC-16(1967), S.764-773.

[62] Shedler,G.S. und Yang,S.C.: Simulation of a Model of Paging System Performance. IBM Syst.Journ.(1971), S.113-128.

[63] Campbell,G.,Fuchel,K. und Heller,S.: The Use of Extended Core Storage in a Multiprogramming Operating System. Aus: Software Engineering. Computer and Information Sciences, Vol.1. Academic Press. New York/London, 1970, S.79-89.

[64] Courtois,P.J. und Georges,J.: An Evaluation of the Stationary Behavior of Computations in Multiprogramming Computer Systems. Intern.Comp.Symp. Bonn, 21.-22.Mai 1970, S.98-113.

[65] Spiess,P.P.: A Queueing Model Analysis of the Multiplexed Use of a Central Processor Unit andan I/O-Channel. Intern.Comp.Symp. Bonn, 21.-22.Mai 1970, S.282-299.

[66] Spiess,P.P.: Analyse eines Warteschlangenmodells für Teilnehmer-Rechensysteme. Elektron. Rechenanl. 12(1970), H.6, S.306-314.

[67] Swoboda,J.: Verkehrsfragen innerhalb einer Rechenanlage. Elektron. Rechenanl. 12(1970), H.5, S.249-252.

[68] Hofmann,J.: Simulation des Verkehrs zwischen Rechnerkernen und einem verschränkten Arbeitsspeicher. Elektron. Rechenanl. 12(1970), H.5, S.253-258.

[69] Boland,L.J.,Granito,G.D.,Marcotte,A.U.,Messina,B.U. und Smith,J.W.: The IBM System/360 Model 91: Storage System. IBM Syst.Journ.(1967),S.54-68.

[70] Flores,I.: Derivation of a Waiting-Time Factor for a Multiple-Bank Memory. J. ACM Vol.11(1964), No.3, S.265-282.

[71] Swoboda,J.: Zur Wirksamkeit von "look ahead". Fachtagung Rechnerstrukturen und Betriebsprogrammierung. Erlangen, Oktober 1970.

[72] Swoboda,J.: Sprachenorientierte Rechner - Probleme der Adressierung. Elektron. Rechenanl. 12(1970), H.1, S.26-35.

[73] Shemer,J.E. und Gupta,S.C.: A Simplified Analysis of Processor "Look-Ahead" and Simultaneous Operation of a Multi-Module Main Memory.IEEE Transact.Comp. Vol.C-18(1969), S.64-71.