FORTSCHRITT-BERICHTE VDI

Dipl.-Inform. Jürgen Dunkel, Bochum

Modellierung und Analyse lastinduzierter Speicherfehler

Reihe **10**: Informatik/
Kommunikationstechnik Nr. **134**

VDI VERLAG
Verlag des Vereins Deutscher Ingenieure · Düsseldorf

Dunkel, Jürgen

Modellierung und Analyse lastinduzierter Speicherfehler

Fortschr.-Ber. VDI Reihe 10 Nr. 134. Düsseldorf: VDI-Verlag 1990.
172 Seiten, 37 Bilder, 11 Tabellen.

Für die Dokumentation: Speicherfehler — Performability — Lastinduktion — Paging — Dekomposition — Iterative Verfahren — Warteschlangen-Netze — Working — Set Prinzip — Markovmodelle — Erneuerungstheorie

Ein sehr großer Anteil der in Rechensystemen auftretenden Fehler ereignet sich im Speicher. In dieser Arbeit wird ein zerlegungsorientiertes Modell entwickelt, das die Wechselwirkungen zwischen Speicherfehlern und Systemleistung untersucht. Zunächst wird das Speicherverhalten eines Auftrags durch ein mehrphasiges Independent-Reference-Modell charakterisiert. Dies dient als Grundlage eines Modells zum Auftreten von Störungen, in das Lastcharakteristika wie die Auftrags-Verweildauer, die Seitenzugriffs-Rate und die Paging-Rate eingehen. Anschließend kann die Wahrscheinlichkeit, mit der ein Speicherfehler entdeckt wird, berechnet werden. Die zur Behandlung von Speicherfehlern erforderlichen Maßnahmen bestimmen die mittlere durch Fehler induzierte Last. Die Wechselwirkungen zwischen Fehler- und Leistungsverhalten werden durch ein System nichtlinearer Gleichung beschrieben, für dessen Lösung ein iteratives Verfahren abgeleitet wird. Abschließend wird mit ausführlichen Beispielen das Modell erläutert und der Einfluß einiger Modell-Parameter auf Leistungs- und Zuverlässigkeitskenngrößen untersucht.

ISSN 0178-9627

ISBN 3-18-143410-8

Danksagung

Herrn Prof. Dr. W. Schneeweiß möchte ich für die wissenschaftliche Betreueung meiner Dissertation, für seine zahlreichen Hinweise und Anregungen herzlich danken.
Der gleiche Dank gilt auch Prof. Dr. M. Dal Cin für seine Bereitschaft, das Zweitgutachten anzufertigen.
Ganz herzlich möchte ich mich an dieser Stelle auch bei meinem Kollegen Dr. H. Bähring für die zahlreichen Diskussionen sowie für hilfreiche Hinweise bedanken.

0 Inhalt

NOTATION

a_i	Aufenthaltswahrscheinlichkeit der Seite s_i im Hauptspeicher
B^G	Gesamt-Bedienzeitanforderung eines Auftrags
B, B_I	Bediendauer des Auftrags in Phase I
B^K	Bedienzeit zwischen zwei Konsistenzchecks
B^R_i	die bei einer Fehlerentdeckung in Seite s_i zu wiederholende Bearbeitungsdauer
B^R	die gesamte durch Fehler verursachte zusätzliche Bediendauer eines Auftrags
C_i	Anzahl der Fehlerüberprüfungen von s_i
D^E_i	Einlagerungsdauer der Seite s_i
D^R_i	Inter-Referenz-Zeit der Seite s_i
D^K_i	Inter-Konsistenzcheck-Zeit der Seite s_i
$e(k)$	Wahrscheinlichkeit, mit der von einer einzelnen Störung k Seiten betroffen sind
$\bar{e}$	mittlere Anzahl der von einer einzelnen Störung betroffenen Seiten
G	Normierungskonstante der Lastvektoren
H	Anzahl der Seiten im Hauptspeicher
I, J	Phasenindizes
K	Anzahl der speicherresidenten Aufträge
$\mathbf{K} = (k_1 \ldots, k_L)$	Lastvektor
k_I	Anzahl der Aufträge aus Klasse/Phase I
L	Anzahl der Phasen bzw. Lastklassen
$m(I)$, m	Anzahl der Seiten in einer Phase I
M	Größe des gesamten virtuellen Adreßraums (in Seiten)
$^{(n)}$	Index für den n-ten Iterationsschritt
N^F_i	Anzahl aller bei s_i entdeckten Fehler
N^P	Anzahl der Paging—Fehler

N^{U_i}	Anzahl ereignisunabhängiger Störungen bei Seite s_i
N^{Z_i}	Anzahl zugriffsinduzierter Störungen bei Seite s_i
p^{CF}, p^{CP}	Übergangswahrscheinlichkeiten für die Übergänge von der CPU zur File- bzw. zur Paging-Station
$Pr^U(i,t)$	Wahrscheinlichkeit, mit der im Zeitraum t genau i ereignisunabhängiger Störungen auftreten
$Pr^Z(k,Z)$	Wahrscheinlichkeit, daß bei insgesamt Z Zugriffen k Störungen auftreten
p_a	Wahrscheinlichkeit, mit der eine Störung zur Verfälschung einer Seite führt
p_z	Wahrscheinlichkeit, mit der ein Seitenzugriff zu einer Störung führt
p_p	Wahrscheinlichkeit, mit der ein Seitenwechsel zu einer Störung führt
Prob[a]	Wahrscheinlichkeit des Ereignisses a
$P_{k_1..k_L}$	$Prob[\mathbf{K} = (k_1,k_2,\ldots,k_L)]$
$\mathbf{P} := [p_{IJ}]$	Übergangsmatrix des Phasenwechsels
q^{U_i}	Wahrscheinlichkeit, daß bei einer Fehlerüberprüfung von s_i ein ereignisunabhängiger Fehler entdeckt wird
q^{Z_i}	Wahrscheinlichkeit, daß bei einer Fehlerüberprüfung von s_i ein zugriffsinduzierter Fehler entdeckt wird
q_i	Wahrscheinlichkeit, daß bei einer Fehlerüberprüfung von s_i ein Fehler entdeckt wird
$\mathbf{Q} = [q_{ij}]$	Übergangsmatrix des Seitenzugriffs
$\mathbf{Q}^*$	Übergangsmatrix, in Blockdiagonalform transformiert
$\mathbf{Q}_I$	Teilmatrix einer Phase I
$R^Y(t)$	Störungsfreiheit eines Auftrags
$\bar{s}$	mittlerer Erneuerungsabstand
T	Fenstergröße (in Seiten)

V_i	Verweildauer der Seite s_i im Hauptspeicher
V, V_I	Gesamtverweildauer des Auftrags in Phase I
$w(T)$	Arbeitsmengengröße bei Fenstergröße T
X^Z, $Z \varepsilon \{U,Z,P\}$	Zufallsvariablen für die Zeitdauern, in denen ein Auftrag keine (ereignisunabhängige, zugriffsinduzierte, bzw. durch Seitenwechsel verursachte) Störung erleidet
Z_i	Anzahl der Zugriffe auf Seite s_i
Z^C_i	Anzahl der Zugriffe auf s_i zwischen zwei Fehlerüberprüfungen
β_i	Zugriffsrate auf Seite s_i
β, β_I	Seiten-Zugriffsrate für Phase I
Γ	Gesamtzahl aller Seitenwechsel eines Auftrags
Γ^C	Anzahl der zwischen zwei Fehlerüberprüfungen referenzierten Seiten
Γ^R	Anzahl aller durch Fehler verursachten zusätzlichen Seiteneinlagerungen
Γ^P	Anzahl der durch Paging verursachten Seitenwechsel (bei der fehlerfreien Auftragsausführung)
δ_s	Rate der ereignisunabhängigen Störungen
π_i, $\pi_{i(I)}$	Zugriffswahrscheinlichkeit auf die Seite s_i (aus Phase I)
$\boldsymbol{\pi}_I=(\pi_{1(I)},..,\pi_{m(I)})$	Vektor der stationären Zugriffswahrscheinlichkeiten auf die Seiten aus Phase I
π_I	Wahrscheinlichkeit für einen Zugriff auf eine Seite aus Phase I
$\boldsymbol{\pi}=(\pi_1,..,\pi_L)$	Vektor der stationären Phasen-Wahrscheinlichkeiten
μ^{CPU}	Bedienrate der CPU
μ^P, $\mu^{I/O}$	Bedienraten der Paging— bzw. I/O-Station
Φ^F_i	Rate der bei Seite s_i entdeckten Fehler
Φ^F	Gesamt-Fehlerrate
Φ^{HF}	Rate der durch Hauptspeicherfehler induzierten zusätzlichen Seiteneinlagerungen

Φ^{PF}	Rate der Paging-Fehler
$\Phi^{I/O}$	Rate der I/O-Operationen
Φ^{P}_{I}	Paging-Rate in Phase I
$\Omega = \{s_1, s_2, \ldots, s_M\}$	virtueller Adreßraum
$*$	Faltungsoperator

1 Einleitung

Den Wechselwirkungen zwischen Leistung und Zuverlässigkeit von Rechensystemen (**Performability**) wird seit einigen Jahren verstärkt Bedeutung beigemessen /MEYE 80/, /KHMA 86/. Die meisten Arbeiten betrachten dabei das Leistungsverhalten von **fehlertoleranten Systemen** /DALC 79/ mit mehreren Bedienern, die ausfallen und repariert werden können /MUEL 88/, /TRIV 85/, /GAKE 79/, /HUSL 81/, /MUNA 83/, /BEAU 78/. Zur Modellierung werden meist zwei Modelle benutzt, die bspw. nach dem Dekompositionsverfahren von Courtois /COUR 77/ verknüpft werden.
Das erste Modell (`structure model`) untersucht, z.B. mit Markov-Prozessen, wie sich die Systemkonfiguration durch Ausfälle und Reparaturen ändert. Im anderen Modell (`reward model`) wird das Leistungsverhalten jeder Degradationsstufe z.B. mit Hilfe von BCMP-Netzen /BCMP 75/ analysiert.

In dieser Arbeit wollen wir nicht - wie in der genannten Literatur - die Performability von fehlertoleranten Mehrprozessor-Systemen, sondern die von Speicherverwaltungs-Systemen untersuchen. D.h. hier wird nicht der Ausfall von Bedieneinheiten, sondern das Auftreten von **Fehlern im Speicher** betrachtet.

1.1 Zur Bedeutung lastinduzierter Speicherfehler

Bei der **Leistungsbewertung** von Rechensystemen mit seitenorientierter, virtueller Speicherverwaltung ist der Speicher der zentrale Flaschenhals des Systems /LAVE 83/, /GEMI 80/. Der den Aufträgen zur Verfügung gestellte Speicherplatz bestimmt die Paging-Rate und damit auch charakteristische Leistungsindizes wie den Durchsatz und die Antwortzeit.

Aus empirischen Untersuchungen realer Rechner geht hervor, daß aber auch die **Zuverlässigkeit** von Rechnern in starkem Maße vom Speicher abhängt. So werden bspw. in /SKMM 78/ 50 bis nahezu 100% der Systemzusammenbrüche auf Speicherfehler zurückgeführt. Ein wichtiges Ergebnis dieser Untersuchungen ist, daß auch Speicherfehler **lastinduziert** sind, d.h., daß mit steigender Systemlast die Wahrscheinlichkeit eines Fehlers zunimmt. So wird in /IYBM 82/ eine starke Abhängigkeit zwischen Systemfehlern und Paging-Rate festgestellt.

Es erscheint also sinnvoll, Modelle für das Auftreten und die Behandlung von Speicherfehlern zu entwickeln. Doch zunächst soll die Bedeutung transienter, lastinduzierter Speicherfehler genauer erläutert werden.

Lastinduktion

Es ist offensichtlich, daß Fehler mit Ansteigen der Systemlast häufiger auftreten, denn Störungen führen nur dann zu Fehlern, wenn das gestörte Betriebsmittel auch benutzt wird /SCHO 86/. Auch die Ursachen von Störungen können davon abhängen, wie oft die betroffene Komponente beansprucht wird. Wird der Speicher z.B. so betrieben, daß nur beim Zugriff die volle Betriebsspannung anliegt (power down, stand by mode), dann ist die für transiente Fehler in Halbleiterspeichern mitverantwortliche thermische Belastung von der Zugriffsrate abhängig.

Es gibt viele **empirische Untersuchungen**, die eine Lastabhängigkeit von Systemfehlern nachweisen. In /CASI 81/, /CASI 82/ wird angenommen, daß die Wahrscheinlichkeit eines Systemfehlers von dem Zeitanteil abhängt, den sich das Betriebsystem im System-Modus (kernel mode) befindet, denn nur dann wirken sich Fehler nicht nur auf Anwenderaufträge, sondern auch auf für das System lebenswichtige Dienste aus. Weil der System-Overhead mit der Last steigt, ist auch die Zeitdauer im kernel mode lastabhängig.
Selbst wenn zur Beschaffung empirischer Daten sehr genaue Hardware-Monitore benutzt werden, lassen sich oft die exakten Gründe für einen Systemfehler nicht feststellen. D.h. ein funktionaler Zusammenhang zwischen Fehlern und Systemtätigkeit kann nur schlecht abgeleitet werden.

Bei **analytischen Modellen** werden in der Regel nur zeit- und lastunabhängige Fehlerprozesse mit exponential verteilten Fehlerabständen betrachtet.
In einigen Arbeiten über Mehrprozessor-Systeme mit sanftem Leistungsabfall (gracefully degrading systems) /MUNA 83/, /GAKE 79/, /HUSL 81/, /MEWE 88/ wird untersucht, welche Last durch die aktuelle Degradationsstufe noch bearbeitet werden kann. Zugrunde gelegt wird jeweils ein Markovmodell mit einem zweidimensionalen Zustandsraum. Eine Zustandskomponente bezeichnet die aktuelle Systemkonfiguration, die sich gemäß entsprechender Ausfall- und Reparatur-Raten ändert. Die andere Komponente ist ein Maß für die vom System zu bearbeitende Last, deren Verhalten durch Ankunfts- und Bedienraten bestimmt ist. In /HUSL 81/ werden die Zustände des Modells, in denen die aktuellen Lastanforderungen von der gegebenen Systemkonfiguration noch verarbeitet werden können, als modes of operation bezeichnet.

Speicherfehler

Man kann nun untersuchen, mit welcher Wahrscheinlichkeit bei bestimmten System-Komponenten lastinduzierte Fehler auftreten. Eine sehr wichtige Rolle spielen dabei Fehler im Hauptspeicher, denn Speicher sind meist diejenigen Rechnerkomponenten mit der größten Integrationsdichte. Die wichtige Bedeutung von **Hauptspeicherfehlern** ist in vielen empirischen Arbeiten /SKMM 78/, /CASI 82/, herausgestellt worden.
Ebenfalls sehr fehleranfällig ist das **Einlagern von Daten** aus dem Hintergrundspeicher in den Hauptspeicher (**Paging-Fehler**) /HSIT 87/, /BOME 86/.

Ein großer Teil der auftretenden Speicherfehler ist **transient**, d.h. nur von begrenzter Dauer /AVIZ 76/. Solche Fehler werden durch temporäre Ereignisse oder durch externe Störungen verursacht und können behoben werden, indem man ihre Dauer abwartet und dann die betroffenen Programme und Daten in einen fehlerfreien, konsistenten Zustand zurücksetzt (**rollback, checkpointing**) /RAND 75/.
Transiente Fehler sind in wesentlich stärkerem Maße für Systemfehler verantwortlich als permanente Fehler. In /SISW 82/ wird angegeben, daß 80% aller elektronischen Fehler in Rechnern **intermittierender** Art sind, d.h. sie treten nur bei bestimmten Ereignissen auf. Dort wird die MTBE (mean time between errors) eines 1M x 37 bit großen Speichers in MOS-Technologie mit etwa 100 Stunden angegeben, die entsprechende MTBF (mean time between failures) jedoch mit etwa 1500 Stunden. Insgesamt wird festgestellt, daß transiente Fehler ungefähr 10 mal häufiger auftreten als permanente.

Trotz ihrer hohen Auftrittshäufigkeit und immensen Bedeutung werden in der Literatur Speicherfehler meist nicht modelliert. Eine Ausnahme bilden die genannten empirischen Arbeiten. So wird bei Iyer et. al. /IYBM 82/ mit Hilfe von Regressionsanalysen aus Meßdaten eines IBM OS/VS2 Systems eine starke Abhängigkeit zwischen den Systemfehlern und der Paging-Rate festgestellt.

Bei analytischen Modellen bleiben Speicherfehler meist unberücksichtigt oder sie werden, wie in /SCHO 86/, nur grob mit Hilfe des Gesamtspeicherbedarf eines Auftrags und der Fehlerrate pro Speichereinheit abgeschätzt. Es gibt jedoch keine Arbeit, in der differenzierter das **auftragsspezifische Speicherverhalten (program behaviour)** /SPIR 77/ zur Untersuchung von Zuverlässigkeitsfragen herangezogen wird.

1.2 Ziele der Arbeit

Im Gegensatz zu anderen Arbeiten auf dem Gebiet der "**Performability**" werden hier speziell **Speicherfehler** betrachtet. Dabei soll einerseits untersucht werden, welchen Einfluß Leistungsgrößen wie Paging-Raten und Verweildauern auf das Auftreten von Speicherfehlern haben. Andererseits wird die durch Speicherfehler verursachte Leistungsminderung in Betracht gezogen.
In unser Modell soll zum einen die **Hardware-Struktur** des Systems eingehen, z.B. die Organisation der Speicherchips und die Hauptspeichergröße. Weiterhin gilt es aber auch, die Architektur des **Betriebssystems** zu berücksichtigen, d.h. die Algorithmen zur Speicherverwaltung (z.B. Wahl der Fenstergröße, vgl. Kapitel 2) und die Mechanismen zur Fehlerentdeckung und -behandlung.

Bei unserer Untersuchung werden Rechensysteme mit **virtuellen seitenorientierten Speichern** zugrunde gelegt. Das Speicherverhalten von Aufträgen wird in Kapitel 2 modelliert. Dabei können wir uns teilweise an bei der Leistungsbewertung übliche Beschreibungsmodelle des Programmverhaltens /SPIR 77/ anlehnen. Zusätzlich gilt es, Einflußgrößen von Speicherfehlern zu bestimmen.

In Kapitel 3 werden die grundlegenden Probleme bei der **Entwicklung von Performability-Modellen** aufgezeigt und eine Zerlegung in Teilmodelle vorgenommen.

Um die mit Speicherfehlern verbundenen Abläufe zu untersuchen, werden in den Kapiteln 4 und 5 die folgenden Teilmodelle betrachtet:

I **Modelle des Auftretens von Störungen** (Kapitel 4)
Es werden Modelle entwickelt, die das Auftreten von Störungen im Speicher abbilden. Weil Störungen lastinduziert sind, gehen Beschreibungsgrößen der Systemlast als Parameter in die Modelle ein. In diesem Kapitel wird ebenfalls berücksichtigt, wie die Auswirkung einer einzelnen Störung von der Hardware-Organisation des Speichers abhängt.

II **Modelle der Fehlerentdeckung** (Kapitel 5)
In diesen Modellen wird unterschieden, zu welchen Zeitpunkten Speicherfehler entdeckt werden. Bei drei verschiedenen Verfahren der Fehlerentdeckung untersuchen wir, mit welcher Wahrscheinlichkeit Speicherseiten einen Fehler erleiden. Jedes der Verfahren erfordert unterschiedliche Maßnahmen der Fehlerbehandlung.

III **Modelle der Fehlerbehandlung** (Kapitel 5)
Es muß bestimmt werden, welche Auswirkungen auftretende Fehler auf das System haben. Wir werden insbesondere feststel-

len, welche zusätzliche Last durch die Fehlerbehandlung entsteht. Diese wirkt sich wiederum auf die Leistungsgrößen des Systems aus.

In **Kapitel 6** wird ein **approximatives Warteschlangenmodell** zur Bewertung des Leistungsverhaltens angegeben. Die so berechenbaren Leistungsindizes dienen als Grundlage unseres Fehlermodells.

In **Kapitel 7** werden schließlich, die in den Kapiteln 4 bis 6 entwickelten **Leistungs- und Fehlermodelle** in einem Gesamt-Modell **integriert**. Um die wechselseitigen Abhängigkeiten von Zuverlässigkeits- und Leistungsgrößen berücksichtigen zu können, wird dazu ein iteratives Verfahren vorgestellt. Anschließend wird in Abhängigkeit von seiner Bedien- und Verweildauer die Störungsfreiheit eines Auftrags bestimmt, d.h. die Wahrscheinlichkeit, mit der er keinen Speicherfehler erleidet.

In **Kapitel 8** wird das iterative Verfahren ausführlich an einem Beispiel erläutert, und die Auswirkung einiger Modell-Parameter auf Leistungs- und Zuverlässigkeits-Kenngrößen untersucht.

Zusammenfassend wollen wir feststellen, daß zur Untersuchung von Speicherfehlern ein **hierarchisches, hybrides** Modell entwickelt wird, das die verschiedenen Sytemaktivitäten abbildet. Es ist in entsprechende Teilmodelle zerlegt, die mit Hilfe von Gesetzmäßigkeiten der **Wahrscheinlichkeitstheorie**, der **Erneuerungstheorie** und der **Warteschlangen-Theorie** analysiert werden.

Das Ergebnis ist ein Modell, das über die rein statische Analyse von Speicherfehlern, wie sie z.B. in /SCSE 82/ diskutiert wird, weit hinausgeht.

2 Zum Auftreten von Fehlern in Speicherverwaltungs-Systemen

Um Fehler in Speicherverwaltungs-Systemen analysieren zu können, muß zunächst ein Modell für das System und insbesondere für das Speicherverhalten von Aufträgen entwickelt werden. Dazu definieren wir in Abschnitt 2.1 die von uns betrachteten Systeme und stellen einige bei der Leistungsanalyse von Rechensystemen übliche Beschreibungsmodelle des Programmverhaltens vor. In Abschnitt 2.2. werden dann Fehler in Speichern betrachtet, und deren Einflußgrößen bestimmt.

2.1 Systemsicht und Programmverhalten

In dieser Arbeit wird von Systemen der folgenden Art ausgegangen:

- Der Speicher besteht aus Haupt- und Hintergrundspeicher, d.h. er besitzt eine hierrarchische Struktur. Jeder Prozessor besitzt eine eigene **virtuelle, seitenorientierte Speicherverwaltung** /DUNK 89/.

- Die Speicherverwaltung arbeitet nach dem **demand paging** Prinzip /DENN 68/, d.h. eine Seite wird erst dann in den Hauptspeicher eingelagert, wenn ein Programm auf ein Datum in dieser Seite zugreifen will.

Die folgende Abbildung 2.1 zeigt schematisch die Struktur eines zweistufigen, virtuellen Seitenspeichers. Darin kennzeichnen die schraffierten Felder die von einem bestimmten Auftrag belegten Speicherseiten.

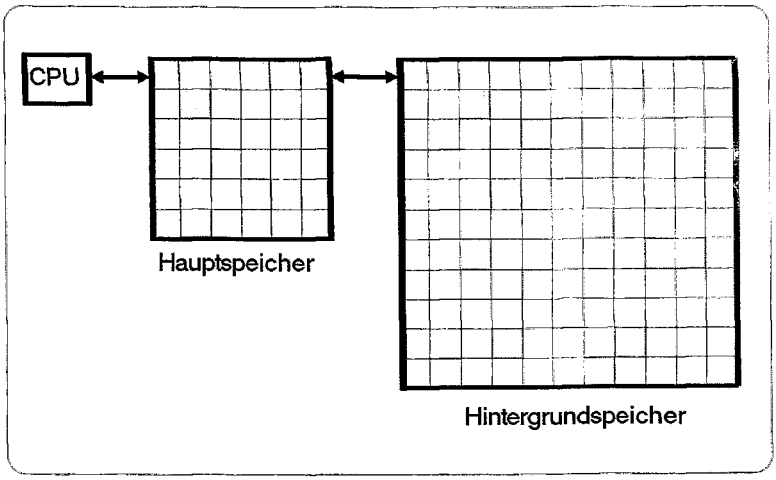


Abb. 2.1: Hierarchischer, zweistufiger Seitenspeicher

In den meisten Betriebssystemen mit **Mehrprogrammbetrieb** erfüllen die Speicherverwaltungssysteme die genannten Voraussetzungen. Unterschiedlich sind allerdings die verschiedenen Strategien, mit denen der Hauptspeicherplatz auf die konkurrierenden Aufträge aufgeteilt wird. Grundlage aller Speicherzuteilungs-Algorithmen ist eine wesentliche Eigenschaft von Programmen, nämlich die sogenannte **Lokalität** /BELA 66/. Die Lokalitäts-Eigenschaft besagt, daß Programme zu einem bestimmten Zeitpunkt nur einen Teil ihres von Programmcode und Daten belegten Speicherplatzes benötigen.
Um exaktere Aussagen formulieren zu können, müssen zunächst einige Bezeichnungen eingeführt werden.

Der **virtuelle Adreßraum** Ω eines Auftrags ist in Seiten unterteilt. Die Menge der Seiten sei:

$$\Omega = \{s_1, s_2, \ldots, s_M\} \quad . \tag{2.1-1}$$

In (2.1-1) ist M die Anzahl aller virtuellen Seiten des Auftrags.
Das Speicherverhalten von Aufträgen wird nicht durch die genauen Zugriffs-Adressen, sondern nur durch die Zugehörigkeit der Adressen zu bestimmten Seiten charakterisiert. Das Speicherverhalten läßt sich dann durch die **Seiten-Zugriffsfolge** $\mathbf{r}$ **(page reference string)** beschreiben:

$$\mathbf{r} = (r_1, r_2, r_3, \ldots, r_t, \ldots) \quad , \quad \text{mit } r_t \in \Omega \; . \tag{2.1-2}$$

Dabei meint r_t den t-ten Speicherzugriff und $r_t=s_i$, falls dieser einer Adresse der Seite s_i gilt.

Die Lokalitäts-Eigenschaft läßt sich bei den Zugriffsfolgen daran erkennen, daß zeitlich zusammenliegende Zugriffsmuster sehr ähnlich, zeitlich weit auseinanderliegende jedoch fast unabhängig voneinander sind. Mit anderen Worten, die Zugriffswahrscheinlichkeit auf eine bestimmte Seite ändert sich nur sehr langsam, d.h. sie erscheint nahezu stationär.

Speichervergabe-Strategien

Wie in /DEKA 75/ beschrieben, kann zwischen Speichervergabe-Strategien unterschieden werden, die jedem Programm eine feste oder eine variable Anzahl von Seiten zugestehen (fixed/variable space policy). Wegen der Lokalitäts-Eigenschaft mit oftmals sehr unterschiedlichem Speicherbedarf innerhalb eines Programms ist eine variable Speicherbelegung günstiger /DESC 72/ und wird auch in fast allen Betriebssystemen angewendet. Die meisten Verfahren orientieren sich am von Denning /DENN 68/ eingeführten **working set model**: für jedes Programm sind diejenigen Seiten eingelagert, die bei den letzten Zugriffen angesprochen wurden.

Definition:
Sei $\mathbf{r} = (r_1, r_2, r_3, \ldots, r_t)$ die Zugriffsfolge zum Zeitpunkt t eines Auftrags, der insgesamt M verschiedene Seiten benutzt. Dann ist die **Arbeitsmenge** (**working set**) W(t,T,M) definiert als die Menge der verschiedenen Seiten, die in der Zugriffs(teil)folge $r_{t-T+1}, \ldots, r_t$ auftreten.

Die **Arbeitsmengengröße** w(t,T,M) ist die Anzahl der in der Arbeitsmenge zum Zeitpunkt t enthaltenden Seiten. Im folgenden werden wir meist die mittlere Arbeitsmengengröße betrachten, die wir mit $\overline{w}(T)$ bezeichnen werden.
Dabei wird T die **Fenstergröße** genannt. Die Wahl eines geeigneten T hat großen Einfluß auf das Leistungs- und Fehlerverhalten eines Rechensystems.

Offensichtlich ist die Arbeitsmenge eines Auftrags nicht konstant, sondern ändert sich in Abhängigkeit von seinem Speicherbedarf. Deshalb kann man sogenannte **Phasen** /DEKA 75/ oder **Lokalitätsbereiche** /COUR 77/ unterscheiden, in denen ein Auftrag unterschiedliches

Speicherverhalten aufweist. Einzelne Auftrags-Phasen können z.B. mit bestimmten Programm-Moduln korrespondieren.

Wird von einem Programm eine Seite benötigt, die sich nicht im Hauptspeicher befindet, so liegt ein sogenannter **Seitenfehler (page fault)** vor, und die betreffende Seite muß in den Hauptspeicher eingelagert werden (**paging**). Der Begriff des "Fehlers" wird hier natürlich nicht im zuverlässigkeitstheoretischen Sinn verstanden, sondern meint das Fehlen einer erforderlichen Seite im Hauptspeicher. Ist der Hauptspeicher bereits vollständig belegt, so muß eine der eingelagerten Seiten ihren Platz räumen. Es erfolgt ein sogenannter **Seitenwechsel**.

Die **Seitenfehlerwahrscheinlichkeit** gibt die Wahrscheinlichkeit an, mit der auf eine nicht in der Arbeitsmenge befindliche Speicherseite zugegriffen wird. Die Rate, mit der Seiten in den Arbeitsspeicher eingelagert werden müssen, wird **Seitenfehlerrate (page fault rate)** genannt.

In /SPIR 77/ wird die Auswirkung der Fenstergröße T auf die mittlere Arbeitsmengengröße $\overline{w}(T)$ und die Seitenfehlerwahrscheinlichkeit f(T) untersucht (vgl. Abbildung 2.2).

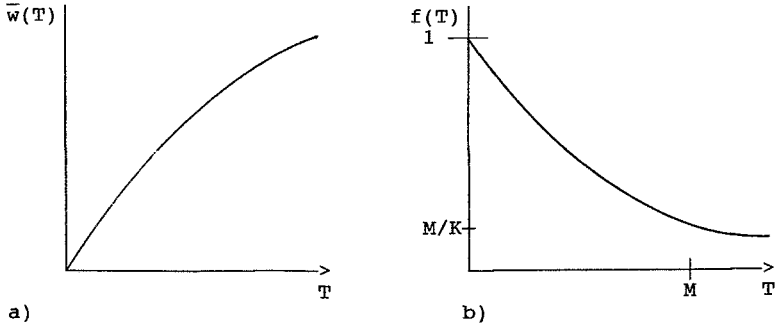


Abb. 2.2: Zusammenhang zwischen Fenstergröße und
a) Arbeitsmengengröße
b) Seitenfehlerrate

Beide Abbildungen beschreiben den Zusammenhang zwischen den betrachteten Größen nur grob. So sind die Arbeitsmengen- und die Fenstergröße offensichtlich nur für ganze Zahlen definiert. Die, in der Literatur übliche, durch Interpolation erhaltene, stetige Kurve veranschaulicht aber besser den konkaven Verlauf, d.h eine Verdopplung der Fenstergröße T bewirkt weniger als eine Verdopplung der

Arbeitsmengengröße. Die Seitenfehlerwahrscheinlichkeit f(T) ist die Ableitung von w(T) /SPIR 77/.

Wenn der virtuelle Adreßraum eines Auftrags die Größe M hat, und die betrachtete Zugriffsfolge eine Länge von K>M aufweist, dann gibt es wenigstens M Seitenfehler, nämlich zur erstmaligen Einlagerung jeder Seite. Als Mindest-Seitenfehlerrate kann somit der Wert M/K erreicht werden.
Man erkennt anhand von Abbildung 2.2 deutlich die Auswirkungen der gewählten Fenstergröße T auf das Systemverhalten.
Bei zu kleinem T ist die Arbeitsmenge so klein, daß sie nicht die Mindest-Menge der Seiten umfaßt, die während einer Phase ständig vom Auftrag benutzt werden, den sogenannten **parachor** /BEKA 69/. Die Folge ist ein starkes Anwachsen der Seitenfehlerrate (**thrashing**), so daß das System unter großem Durchsatzverlust fast nur noch mit Seitenwechseln beschäftigt ist.
Bei zu großer Fenstergröße T sind mehr als die vom Auftrag benötigten Seiten eingelagert, und die Seitenfehlerrate kann nicht mehr weiter verringert werden. Dadurch wird der Hauptspeicher schlecht genutzt, denn es können entsprechend weniger Aufträge in den Speicher eingelagert werden, und der Systemdurchsatz sinkt.

Um das Speicherverhalten von Programmen (program behavior) zu charakterisieren, gibt es verschiedene stochastische Modelle, die dazu dienen, "typische" Zugriffsfolgen zu generieren /CODE 73/, /SPIR 77/, /COUR 77/. Im folgenden werden die drei wichtigsten Modelle beschrieben.

Das IRM Modell (independent reference model)

Dieses Modell geht davon aus, daß man bei gegebener Zugriffsfolge $\mathbf{r} = (r_1, r_2, r_3, \ldots, r_t, \ldots)$ für jede Seite $s_i \varepsilon \Omega$ eine konstante Zugriffswahrscheinlichkeit b_i angeben kann, so daß gilt:

- $b_i := \text{Prob}[r_t = s_i] = \text{const.}$ für alle t, und
- für $i \neq j$ werden die Ereignisse $\{r_t = s_i\}$ und $\{r_t = s_j\}$ als stochastisch unabhängig vorausgesetzt.

(2.1-3)

Es wird also angenommen, daß die Zugriffswahrscheinlichkeit einer Seite stationär, d.h. zeitunabhängig ist, und auch nicht von der zuletzt zugegriffenen Seite abhängt. Dies entspricht natürlich nicht unbedingt der Realität, denn in verschiedenen Lokalitätsbereichen

kann die Zugriffswahrscheinlichkeit auf eine bestimmte Seite s_i sehr unterschiedlich sein. Das IRM-Modell berücksichtigt also nicht in ausreichendem Maße das Phasen-Verhalten von Aufträgen. Es ist aber sehr gut handhabbar, und für dieses Modell liegen die meisten analytischen Ergebnisse vor.

Lifetime functions

In /BEKU 69/ wurden die sogenannten **lifetime functions** l(s) eingeführt. Durch diese Funktionen wird in Abhängigkeit von dem einem Auftrag zur Verfügung stehendem Speicherplatz s (in Seiten) der zeitliche Abstand zweier Seitenfehler, also die MTBP (mean time between page faults) ermittelt. In der Literatur werden unterschiedliche Formen von lifetime functions angegeben, vgl. Abb. 2.3.

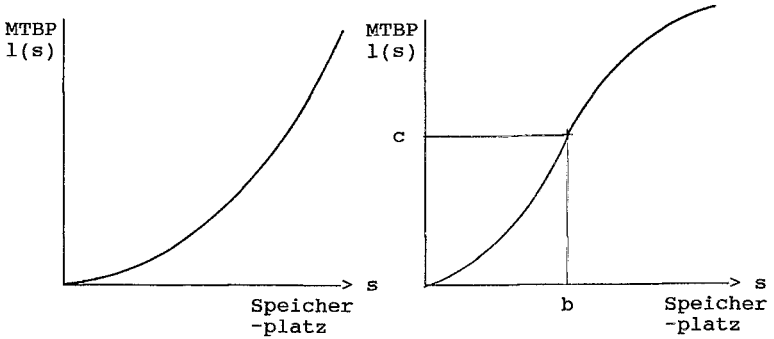


Abb. 2.3 : lifetime functions

Die linke Abbildung zeigt eine von Belady et al. /BEKA 69/ vorgeschlagene Funktion der Form:

$$l(s) = a \cdot s^k \quad , \text{ mit } \quad a > 0 \quad \text{und} \quad 1.5 < k < 2.5 \quad . \tag{2.1-4}$$

Die Funktion in der rechten Abbildung ergibt sich aus einem Ansatz von Chamberlin et al. /CHFL 73/, der Form:

$$l(s) = \frac{2 \cdot b}{1 + (c/s)^2} \quad , \text{ mit } b, c > 0 \quad . \tag{2.1-5}$$

Die Parameter a, b und c sind vom Programm und vom Rechner abhängig und sollen die speziellen Lokalitäts- und Speichereigenschaften widerspiegeln. Gemeinsam ist allen lifetime functions, daß sie mit dem zur Verfügung stehenden Speicherplatz s ansteigen und für kleine s einen konvexen Verlauf aufweisen.

Das MRM-Modell (Markovian Reference Model)

Dieses Modell geht von einer quadratischen Matrix Q aus:

$$Q = [q_{ij}], \quad \text{wobei } q_{ij} := \text{Prob}[r_t = s_j \mid r_{t-1} = si]. \tag{2.1-6}$$

D.h. das Matrixelement q_{ij} gibt die Wahrscheinlichkeit an, mit der zur Zeit t auf die Seite j zugegriffen wird, unter der Bedingung, daß zur Zeit t-1 ein Zugriff auf Seite *i* erfolgte. Ein solches Modell zur Spezifikation des Programmverhaltens wurde zuerst von Aho et al. in /AHDU 71/ eingeführt.

Dekompositions-Approximation bei MRM

Betrachtet man die Übergangsmatrix Q, so läßt sie sich in quadratische Submatrizen Q_I^* zerlegen, die jeweils das Speicherverhalten innerhalb einer bestimmten Phase beschreiben. Die Übergangswahrscheinlichkeiten außerhalb dieser Submatrizen sind vergleichsweise sehr klein.
Es liegt also eine **nearly-completely-decomposable** Matrix (NCD-Matrix) /COUR 77/ der Form

$$Q = Q^* + \alpha \cdot C \tag{2.1-7}$$

vor, wobei

$$Q^* = \begin{bmatrix} Q_1^* & & 0 \\ & Q_2^* & \\ & \blacksquare & \\ & \blacksquare & \\ 0 & & Q_L^* \end{bmatrix} \tag{2.1-8}$$

Dabei ist L die Anzahl verschiedener Lokalitätsbereiche (Phasen) und die Q_I^* mit $I \varepsilon \{1,..,L\}$ sind quadratische Submatrizen der Ordnung m(I), wobei m(I) die Anzahl der in Phase I benutzten Seiten ist. Die 0-Matrizen in (2.1-8) bedeuten, daß alle Matrixelemente außerhalb der Submatrizen Q_I^* den Wert 0 besitzen, d.h. die Matrix Q^* besitzt **Blockdiagonalform.**

In (2.1-7) ist α ein Maß für die maximale Wahrscheinlichkeit, eine Phase zu verlassen. Es errechnet sich aus:

$$\alpha = \max_{1 \le I \le L} \left(\max_{1 \le i(I) \le m(I)} \left(\sum_{J \neq I} \sum_{1 \le j(J) \le m(J)} q_{i(I)j(J)} \right) \right). \qquad (2.1\text{-}9)$$

Die Schreibweise in (2.1-9) ist an /COUR 77/ angelehnt. Darin bezeichnet $q_{i(I)j(J)}$ das Matrixelement q_{ij}, wobei i(I) ein Index aus der Submatrix Q_I^*, sowie j(J) aus Q_J^* sein soll.

Es sei $\pi_I = (\pi_{1(I)}, \ldots, \pi_{m(I)})$ der Vektor der stationären Wahrscheinlichkeiten für einen Zugriff auf eine Seite aus der Phase I. Dann ist:

$$\pi_I := \sum_{1 \le i(I) \le m(I)} \pi_{i(I)} \qquad (2.1\text{-}10)$$

die stationäre Wahrscheinlichkeit in Phase I zu sein. Nach der Theorie von P.J. Courtois /COUR 77/ lassen sich die stationären Zugriffswahrscheinlichkeiten des gesamten virtuellen Speicherraums wie folgt approximieren:

Man bestimmt zuerst für jede Phase den Vektor der stationären Zugriffswahrscheinlichkeiten:
$\pi_I^* = (\pi_{1(I)}^*, \ldots, \pi_{m(I)}^*)$, der sich durch Lösen des Gleichungssystems:

$$\pi_I^* = Q_I^* \cdot \pi_I^* \qquad (2.1\text{-}11)$$

errechnen läßt. Wegen der NCD-Eigenschaft der Transitionsmatrix lassen sich anschließend die gesuchten stationären Zugriffswahrscheinlichkeiten des gesamten virtuellen Speicherraums durch:

$$\pi_{i(I)} = \pi_I \cdot \pi_{i(I)}^* \qquad (2.1\text{-}12)$$

approximieren.

Die Zugriffswahrscheinlichkeiten aller virtuellen Seiten bilden in den Kapiteln 5 und 6 die Grundlage unserer Modellentwicklung, z.B. für die Berechnung von Fehlerraten oder Arbeitsmengengrößen.

Nach diesen allgemeinen Grundlagen seitenorientierter Speicherverwaltungs-Systeme stellen wir im folgenden grundlegende Überlegungen zum Auftreten lastinduzierter Speicherfehler vor.

2.2 Fehler in Speicherverwaltungs-Systemen

In diesem Abschnitt wollen wir die grundlegenden Einflußgrößen von Speicherfehlern bestimmen. Dazu müssen wir zunächst eine exakte **Terminologie** festlegen, um die bei Speicherfehlern auftretenden Phänomene unterscheiden zu können. Wir verwenden in dieser Arbeit die von Laprie /LAPR 85/ eingeführten Begriffe.

- Ursache eines Fehlers ist eine **Störung** (**fault**). Mit Störungen meinen wir hier diejenigen (physikalischen) Ereignisse, die im Speicher abgelagerte Daten verändern. Solche Ereignisse sind z.B. eine Spannungsschwankung, ein Übersprechen auf Daten-Leitungen oder das Auftreffen eines α-Teilchens.

- Eine Störung führt zu einer **Verfälschung** (**error**), d.h. zu einem nicht korrekten Datum (im Speicher). Dabei steht nicht fest, ob eine Verfälschung eine (negative) Auswirkung auf das Systemverhalten besitzt. Manche Verfälschungen im Speicher werden - bspw. durch fehlerkorrigierende Codes - maskiert, andere kommen nicht zum Tragen, weil die gestörten Speicherzellen gar nicht benutzt werden.

- Ein **Fehler** (**failure**) soll im weiteren eine Verfälschung bezeichnen, die mit einem dem System zur Verfügung stehenden Verfahren entdeckt wird, und einer **Behandlungsmaßnahme** bedarf, um ein korrektes Arbeiten des Systems zu gewährleisten. D.h., nur eine nicht maskierbare Verfälschung kann zu einem Fehler führen. Die Fehlerbehandlungsmaßnahme hängt von der Fehlerauswirkung ab.

Die genannten Begriffe lassen sich durch die folgende Abbildung 2.4 verdeutlichen.

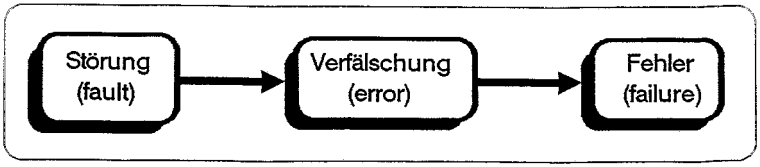


Abb. 2.4: Terminologie zum Auftreten von Fehlern

Grundsätzlich können Fehler in Speichersystemen entweder **im Hauptspeicher** oder beim **Einlagern von Seiten** auftreten. Die letztgenannte Fehler-Art soll im weiteren **Seitenwechsel-Fehler** oder **Paging-Fehler** genannt werden. Sie ist nicht mit den oben eingeführten Seitenfehlern (page faults) zu verwechseln.
Die Ursachen beider Fehler-Arten, ein genaues Fehlermodell, sowie quantitative Aussagen werden in Kapitel 4 angegeben. Wir wollen jetzt jedoch die für die Auftrittshäufigkeit beider Fehler-Arten verantwortlichen, wesentlichen Einflußgrößen ableiten.

Fehler im Hauptspeicher

Wir nehmen an, daß das Risiko, mit dem ein Auftrag einen Hauptspeicherfehler erfährt, im wesentlichen durch zwei Größen bestimmt ist:

- Die Gefahr, daß eine virtuelle Seite von einem Speicherfehler betroffen wird, steigt offensichtlich mit deren **Verweildauer** im Dispatching-System. Von der Ausführungsdauer einzelner Aufträge ist sie hingegen unabhängig. Die Verweildauer von Aufträgen im Dispatching-System hängt von sehr vielen Faktoren ab. Aus Leistungsanalysen ist bekannt, daß sie sowohl von der Systemlast als auch von der den Aufträgen zur Verfügung stehenden Arbeitsmengengröße abhängt. Ein entscheidender Parameter ist ebenfalls die Paging-Rate.

- Das Risiko, daß ein Auftrag einen Hauptspeicherfehler erleidet, ist offensichtlich auch durch die **Größe des** von ihm belegten **Hauptspeicherplatzes** bestimmt. Wie im vorhergehenden Abschnitt beschrieben, ist bei einer variablen Speichervergabe-Strategie (z.B. dem working-set-Prinzip) die Arbeitsmenge nicht konstant, sondern ändert sich dynamisch mit der Auftrags-Ausführung. Insbesondere kann dabei ein Auftrag verschiedene Phasen durchlaufen.

Fehler beim Seiteneinlagern

In unserem Modell machen wir die folgenden Parameter für Paging-Fehler verantwortlich:

- Paging-Fehler können definitionsgemäß nur bei einem Seitenwechsel auftreten. Ihre Häufigkeit hängt also im wesentlichen von der Anzahl der durchzuführenden Seiteneinlagerungen ab, d.h. sie ist direkt proportional zur **Paging-Rate**. Das Auftreten von

Paging-Fehlern ist sicherlich von der Verweil- und Ausführungsdauer des Auftrags unabhängig.

Wechselwirkungen zwischen Leistungs- und Zuverlässigkeitsgrößen

In virtuellen Speicherverwaltungssystemen bestehen eine Vielzahl von **Wechselwirkungen zwischen Leistung** (performance) auf der einen **und Zuverlässigkeit** (dependability) /LAPR 85/ auf der anderen Seite:

- Wie oben erläutert, wirken sich Leistungsgrößen, wie die Verweildauer von Aufträgen oder die Paging-Rate, direkt auf das Fehlerverhalten des Systems aus.

- Zuverlässigkeitsgrößen, wie z.B. Fehler-Raten, beeinflußen wiederum die Systemleistung. Denn jeder aufgetretene Fehler muß behoben werden und erfordert zur Fehlerbehandlung entsprechend zusätzliche Systemaktivitäten. Mögliche Maßnahmen der Fehlerbehandlung sind z.B. eine Wiederholung gestörter Aufträge und die Einlagerung verfälschter Seiten.

Aufgrund dieser Wechselwirkungen sollten Leistungs- und Zuverlässigkeitsanalysen nicht voneinander getrennt durchgeführt werden. Insbesondere hat es auch wenig Sinn, ausschließlich Maße der Zuverlässigkeitstheorie (wie z.B. die Verfügbarkeit) oder nur Leistungs-Maße (wie den Durchsatz) zu betrachten.

Wechselwirkungen zwischen Leistungs- und Zuverlässigkeitsgrößen wurden bisher ausschließlich bei der Untersuchung von **fehlertoleranten Mehrprozessor-Systemen** betrachtet. Deren Analyse führte zur Entwicklung von **Performability**-Modellen /MEYE 80/. Performability ist ein von J.F. Meyer eingeführtes Kunstwort, das sich aus **PERFORM**ance und reli**ABILITY** zusammensetzt und so schon vom Namen her auf den starken Zusammenhang dieser Bewertungsgrößen hinweist. Meyer und andere Autoren /Beau 78/ führten u.a. neue, sogenannte Performability-Maße ein. Ein Performability-Maß ist z.B. die Wahrscheinlichkeit, mit der eine Leistungsgröße, wie die Verweildauer, einen bestimmten Grenzwert nicht übersteigt. Auf formale Definitionen wollen wir an dieser Stelle aber verzichten. (Einen guten Überblick bietet /KHMA 86/.)

Ziel dieser Arbeit ist, die Wechselwirkungen zwischen Leistungs- und Zuverlässigkeitsgrößen in Speichern zu modellieren.

Die folgende Abbildung 2.5 veranschaulicht die Wechselwirkungen der oben genannten Größen. Dabei sei der Einfachheit halber eine bestimmte Phase des Auftrags betrachtet, d.h. die Arbeitsmengengröße und die Verweildauer beziehen sich auf eine einzelne Auftrags-Phase.

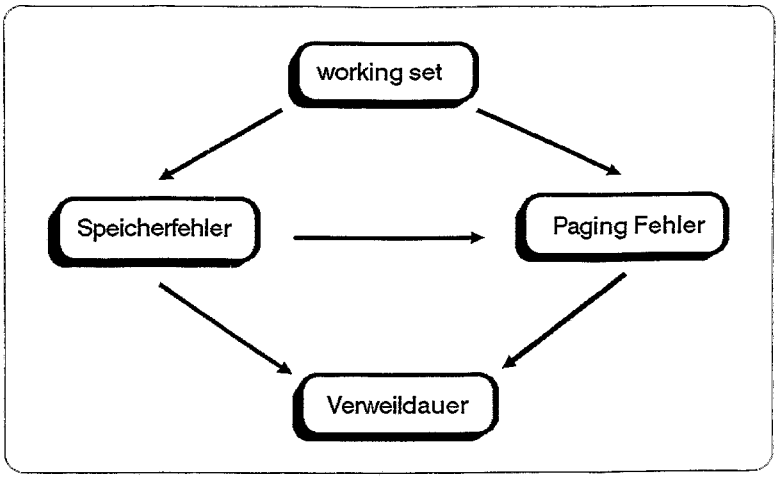


Abb. 2.5 : **Wechselwirkungen zwischen Leistungs- und Zuverlässigkeitsgrößen in Speicherverwaltungs-Systemen.**

Zur Erläuterung von Abbildung 2.5 wollen wir von der Arbeitsmengengröße w(T) des Auftrags ausgehen. Bei einer großen Arbeitsmenge belegt der Auftrag viel Speicher und besitzt ein entsprechend hohes Risiko für einen Hauptspeicherfehler. Bei kleinerer Arbeitsmenge hingegen ist dieses Risiko offensichtlich kleiner, aber dafür steigt, wie anhand der lifetime functions ersichtlich (vgl. Abbildung 2.2), die Seitenfehlerrate, und damit auch die Wahrscheinlichkeit eines Paging-Fehlers.
Jeder Fehler verursacht zusätzliche, zu seiner Behandlung erforderliche Systemlast und verlängert dadurch wiederum die Verweildauer im Dispatching-System. Damit steigt wiederum das Risiko eines Hauptspeicherfehlers. Bei einem erkannten Hauptspeicherfehler ist ein erneutes Einlagern der betroffenen Seite erforderlich und die Wahrscheinlichkeit eines Paging-Fehlers wird ebenfalls erhöht.

Man erkennt also, daß auch bei Speicherverwaltungs-Systemen starke Zusammenhänge zwischen Leistungs- und Zuverlässigkeitsgrößen bestehen. In Abhängigkeit von der Systemlast ändern sich Leistungsgrößen wie die Arbeitsmengengröße und die Verweildauer. Diese bestimmen die Häufigkeit eines Fehler-Auftritts. Und schließlich wirken sich Fehler wiederum auf Leistungsgrößen wie bspw. die Verweildauer aus.

Wie diese Wechselwirkungen prinzipiell in Performability-Modellen untersucht werden können, ist nun Gegenstand des folgenden Kapitels.

3 Zur Entwicklung von Performability - Modellen

In diesem Kapitel wird untersucht, wie eine zerlegungsorientierte Modellierung von Rechensystemen unter Berücksichtigung von Speicherfehlern grundsätzlich erfolgen kann. Dazu werden in Abschnitt 3.1 verschiedene Teilmodelle mit ihren Schnittstellen definiert und spezifische Modellierungssverfahren angegeben. In Abschnitt 3.2 werden einige Überlegungen zu den Wechselwirkungen zwischen den Teilmodellen, sowie zu deren Integration angestellt.

3.1 Modellzerlegung

Performability-Modelle sollen die Wechselwirkungen zwischen Leistung und Zuverlässigkeit widerspiegeln /MEYE 80/, /KHMA 86/. In der Literatur wird dabei in fast allen Arbeiten das Leistungsverhalten von fehlertoleranten Systemen mit mehreren Bedienern, die ausfallen und repariert werden können, untersucht /MUEL 88/, /TRIV 85/, /GAKE 79/, /HUSL 81/, /MUNA 83/, /BEAU 78/. Dabei werden meist zwei Modelle voneinander getrennt betrachtet /SMTR 88/:

- Das **Modell des Ausfall- und Reparaturverhaltens** (`structure state model`) bildet den durch Fehler verursachten Wechsel zwischen verschiedenen Systemkonfigurationen ab. Meist wird zur Modellierung ein (Semi-)Markovprozeß benutzt.

- Das **Modell des Leistungsverhaltens** (`reward model`) untersucht für jede Systemkonfiguration (Degradationsstufe) die Leistungsfähigkeit des Systems. Hier werden meist die in der Leistungsanalyse von Rechensystemen üblichen BCMP-(Warteschlangen-) Netze /BCMP 75/, /LAVE 83/, /GEMI 80/ benutzt.

Beide Modelle sind nach dem - in Abschnitt 2.1 fürs Programmverhalten knapp dargestellten - Ansatz von P.J. Courtois **nearly completely decomposable** (NCD) /COUR 77/ und können deshalb voneinander getrennt untersucht werden. Die NCD-Eigenschaft ist erfüllt, weil die Ausfallraten von Komponenten um mehrere Größenordnungen kleiner sind als die Raten, mit denen Aufträge innerhalb einer Systemkonfiguration bedient werden /SMTR 88/, /MEYE 81/, /DALC 82/.

Durch spezielle Performability-Maße werden die Ergebnisse beider Modelle miteinander verknüpft /MEYE 80/, /KHMA 86/, /MUEL 84/.

In den genannten "klassischen" Arbeiten ändert sich die Systemkonfiguration dynamisch und mit ihr nach dem machine-repairman-Modell /DALC 79/ auch die Ausfallrate.

Performability-Modelle virtueller Speicher

In dieser Arbeit wird eine andere Sichtweise gewählt. Wir wollen ein stochastisches Modell entwickeln, in dem vor allen Dingen die **Systemlast** eine entscheidende Bedeutung besitzt. In Abhängigkeit von der Last, d.h. u.a. vom Speicherverhalten, soll sich die Fehlerwahrscheinlichkeit eines Auftrags ändern. Umgekehrt können Fehler aber auch die Systemlast beeinflussen. Im einzelnen soll unser Performability-Modell die folgenden Eigenschaften aufweisen.

- Es werden ausschließlich **Speicherfehler** betrachtet.
- Die Häufigkeit von Speicherfehlern hängt direkt von verschiedenen **Leistungsgrößen**, z.B. der Verweildauer im Speicher und der Paging-Rate ab.
- Die Auswirkung bestimmter **Betriebssystem-Parameter**, z.B. die Wahl der Fenstergröße, auf Leistungs- und Zuverlässigkeitsmaße soll modelliert werden können. Dazu muß in unserem Modell das **Programmverhalten** /SPIR 77/ detailliert abgebildet werden.
- Das Auftreten von (Speicher-)Fehlern bewirkt eine Veränderung der **Systemlast**. Sie hängt in starkem Maße von der Art der **Fehlerentdeckung** und **-behandlung** ab.
- Das Auftreten von Fehlern verursacht damit auch eine **Veränderung** der **Systemleistung**.

Um den oben genannten Anforderungen gerecht zu werden, und die Komplexität des Gesamt-Modells zu reduzieren, wollen wir die folgenden drei Teilmodelle zu den Problemkreisen

- Leistungsverhalten,
- Auftreten von Störungen,
- Entdeckung und Behandlung von Fehlern

untersuchen.

1. Modell des Leistungsverhaltens

Das **Leistungsmodell** soll zwei Hauptaufgaben erfüllen. Zum einen soll mit üblichen Leistungsindizes, wie z.B. Durchsatz und Antwortzeit, die Leistungsfähigkeit des Systems abgeschätzt werden. Zum anderen liefert es - nach unseren Annahmen aus Abschnitt 2.2 - **Eingabe-Parameter für das Fehlermodell**. Die wichtigsten Parameter sind dabei die Verweildauer eines Auftrags im Speicher und seine Paging-Rate. Diese Leistungsgrößen sind für einen Auftrag nicht nur durch seine eigenen Eigenschaften festgelegt, sondern hängen sehr stark von der aktuellen Systemlast, d.h. der Anzahl der aktiven Aufträge, deren Paging-Raten, usw. ab. So müssen zur Berechnung der Verweildauern offensichtlich die Verzögerungen und Wartezeiten eines Auftrags an verschiedenen Betriebsmitteln, wie z.B. Platten und E/A-Kanälen berücksichtigt werden.
Zur Modellierung des Leistungsverhaltens gibt es verschiedene Ansätze und zahlreiche Arbeiten /BEIL 88/, /FERR 78/, /LAVE 83/, /GEMI 80/, wobei als Modellierungsmethode prinzipiell eine Simulation oder eine analytische Modellbildung möglich ist.

In dieser Arbeit wollen wir uns ausschließlich mit der **analytischen Modellierung** beschäftigen, denn gerade bei sehr selten auftretenden Ereignissen, wie Speicherfehlern, wären extrem lange Simulationsläufe erforderlich. Das in der Leistungsanalyse von Rechensystemen benutzte adäquate Modellierungsverfahren ist ein **Warteschlangen**-Netz mit mehreren Bedienstationen /LAVE 83/. Dazu wird jedes im Dispatching-System vorhandene Betriebsmittel (z.B. CPU, Paging-, I/O-Einheit) durch eine Bedienstation modelliert.

In unser Modell soll insbesondere das **lastabhängig Speicherverhalten** eingehen. Dabei soll u.a. berücksichtigt werden, daß jeder Auftrag verschiedene Lokalitätsbereiche /COUR 77/ bzw. Phasen /DEKA 75/ durchläuft (vgl. Abschnitt 2.1). Diese Forderung erschwert allerdings die Analyse von entsprechenden Warteschlangen-Netzen, weil es dann keine exakte analytische (Produktform-)Lösung /GEMI 80/ mehr gibt (siehe Kapitel 6). In der Literatur zur Leistungsanalyse wurden jedoch mit Hilfe der Dekompositions-Approximation Näherungs-Lösungen angegeben /COUR 77/, /BRBC 77/, /BRAN 75/, /COVA 76/. Ein entsprechendes Modell wird in Kapitel 6 entwickelt.

2. Modell für das Auftreten von Störungen

Sowohl für Hauptspeicher- als auch für Paging-Fehler wird im folgenden Kapitel 4 ein Modell entwickelt, welches das Auftreten von **Störungen** (fault occurence) abbildet. Hauptsächlich soll darin die **Auftritts-Wahrscheinlichkeit einer Störung** bestimmt werden.

Wesentlich in diesem Modell ist, daß Störungen **lastinduziert** auftreten. Wie bereits in Abschnitt 2.2 gefordert, sollen in unser Fehlermodell die Verweildauern der Seiten im Speicher und die Paging-Raten eingehen. Wir werden dabei ereignisunabhängige von ereignisabhängigen Störungen unterscheiden:

Ereignisunabhängige Störungen des Speichers werden nicht durch bestimmte Ereignisse induziert, sondern hängen in erster Linie von der vom Auftrag im Speicher verbrachten Zeitdauer und seinem belegten Speicherplatz ab. Ihr Auftreten wird in Abschnitt 4.1 durch einen **Erneuerungsprozeß** modelliert.
Als **ereignisabhängige Störungen** betrachten wir zugriffsinduzierte und durch Seitenwechsel induzierte Störungen, die ausschließlich bei einem Seitenzugriff, bzw. einem Seitenwechsel auftreten können. Die Anzahl solcher Störungen hängt also von den Paging- und Zugriffs-Raten ab und wird als **binomisch verteilt** angenommen (siehe Abschnitt 4.2).

Zusätzlich muß noch das **Ausmaß** einer einzelnen Störung betrachtet werden, d.h. wieviele Seiten von ihr verfälscht werden. Dieses Ausmaß hängt vom Typ der aufgetretenen physikalischen Störung und der Hardware-Organisation des Speichers ab.

3. Modell für das Entdecken und Behandeln von Fehlern

In diesem Modell soll die **Anzahl der entdeckten Fehler** und die **durch** diese **Fehler induzierte** zusätzliche **Last** bestimmt werden.

In der Terminologie dieser Arbeit führt die Entdeckung jeder Seiten-Verfälschung zu einem **Fehler**. Um die Anzahl entdeckter Verfälschungen bestimmen zu können, muß das Verfahren der Fehlerentdeckung ins Modell eingehen. Wesentlich sind dabei die Zeitpunkte, zu denen eine Fehlerüberprüfung des Speichers durchgeführt wird. Je häufiger die Speicherseiten geprüft werden, umso kürzer ist die **Latenzzeit**, d.h. die Zeitdauer zwischen (physikalischer) Störung und Fehlerentdeckung. In Kapitel 5 werden drei verschiedene Verfahren der Fehlerentdeckung modelliert.

Die erforderlichen Maßnahmen zur **Fehlerbehandlung** (fault handling) richten sich nach der aufgetretenen Latenzzeit: Wird ein Fehler erst sehr spät entdeckt, so hat ein Auftrag meist mit den fehlerhaften Daten bereits gearbeitet, und die Verfälschung kann sich daher schon auf andere Speicherseiten ausgebreitet haben (**error propagation**). Eine mögliche Maßnahme zur Fehlerbehandlung ist die (Teil-) Wiederholung eines Auftrags. Zusätzlich müssen die Originale der im Haupt-

speicher verfälschten Seiten aus dem Hintergrundspeicher eingelagert werden.
Weil uns bei Performability-Untersuchungen auch die Auswirkungen von Fehlern auf die Systemleistung interessieren, muß in diesem Teilmodell die durch einen Speicherfehler induzierte Last festgelegt werden. Die zur Fehlerbehandlung zusätzlich erforderliche Last kann durch eine Erhöhung der Paging-Rate, sowie durch eine Verlängerung der Bediendauer quantifiziert werden.

Plausibilität der Modellzerlegung

Für die Untersuchung der Performability von Systemen mit virtueller Speicherverwaltung ist es natürlich wünschenswert, die drei genannten Teilmodelle voneinander getrennt untersuchen zu können, d.h. eine **Modell-Zerlegung** vorzunehmen /TRGE 83/, /BRAN 85/, /MUEL 88/.
Nach der Theorie von P.J. Courtois /COUR 77/ ist eine Modellzerlegung dann möglich, wenn die untersuchten Teilmodelle "lose gekoppelt" oder **nearly completely decomposable (NCD)** sind.

Im folgenden wollen wir begründen, daß das von uns untersuchte Modell diese Eigenschaft besitzt.

Dazu definieren wir zunächst die folgenden **aggregierten (Makro-) Zustände,** in denen sich ein einzelner Auftrag befinden kann. Jeder dieser Makro-Zustände umfaßt eine Menge vieler "Mikro"-Zustände und korrespondiert mit einem entsprechenden Teilmodell.

Aggregierte Auftragszustände:

- Der Auftrag ist **fehlerfrei,** d.h von keinem Speicherfehler betroffen. Dann durchläuft sein Speicherverhalten verschiedene Phasen (Lokalitätsbereiche). Formal gibt es für jede Auftragsphase I, I=1,..,L, einen Makro-Zustand C_I: In die entsprechende Zustandsmenge fallen alle Zustände eines Auftrags in Phase I, in denen keine Störung aufgetreten ist.

- Sobald eine Seite aus der Arbeitsmenge des Auftrags eine Störung erleidet, die zur Verfälschung eines Speicherwortes führt, geht der betroffene Auftrag in einen **fehlerhaften (Makro) Zustand** C_T, mit T=L+1, über. Durch die Entdeckung der Verfälschung manifestiert sich ein Fehler, der eine **Fehlerbehandlung** erfordert. Solange bei dem Auftrag Maßnahmen zur Fehlerbehebung durchgeführt werden, bleibt er in der Zustandsmenge C_T.

Die aggregierten Zustände liefern also nur darüber eine Aussage, in welcher Phase I sich ein ungestörter Auftrag befindet, bzw. ob er

eine Störung erlitten hat. Befindet sich ein Auftrag in einem bestimmten Makro-Zustand C_I, mit $I \in \{1,.., L, L+1\}$, dann treten innerhalb dieses Zustandes eine Vielzahl von Ereignissen auf. Dazu zählen z.B. die Ankunft und die Bedienung an der CPU oder das Einlagern von Seiten. Diese Ereignisse innerhalb eines Makro-Zustandes werden in Kapitel 6 durch ein Warteschlangen-Netz abgebildet. Der Zustandsraum des Warteschlangen-Modells (dessen genaue Definition bspw. in /LAVE 83/, /BEIL 88/ oder /BOAK 82/ zu finden ist) bildet dann die Menge der einem Makro-Zustand zugeordneten Mikro-Zustände.
Insgesamt liefert also die durch die Makrozustände festgelegte Zustands-Aggregierung eine Partitioinierung des gesamten Zustandsraums.

Damit eine getrennte Untersuchung der Teilmodelle möglich ist, wollen wir als nächstes begründen, daß unser Modell die sogenannte **stiffness-Eigenschaft** besitzt /MIRA 81/.

Nachweis der stiffness-Eigenschaft

In unserem Modell ist die für eine Modellzerlegung wichtige **stiffness-Eigenschaft** /MIRA 81/ gegeben. D.h. bei den verschiedenen Teilmodellen unterscheiden sich die zeitlichen Abstände, in denen modellspezifische Ereignisse auftreten, um mehrere Größenordnungen. Genauer gilt hier:

- Die Rate, mit der Ereignisse im fehlerfreien Zustand (z.B. Bedienung oder Ankunft an der CPU-Station, usw.) stattfinden, ist um Größenordnungen höher als die Rate, mit der Störungen auftreten.

- Die zur Fehlerbehandlung benötigte Zeit ist um Größenordnungen kleiner als die Zeitdauer zwischen zwei Störungen.

Zustands-Klassifikation

Bobbio und Trivedi /BOTR 86/ klassifizieren in Modellen mit der stiffness-Eigenschaft die folgenden Zustandsmengen:

- Die Menge der **schnellen Zustände (fast states)**, die mit mindestens einer großen Abgangsrate verlassen werden.

- Die Menge der **langsamen Zustände (slow states)**, die nur mit kleinen Abgangsraten verlassen werden.

Mit Hilfe der Terminologie von Markovprozessen /FELL 68/ erfüllen dann die Makro-Zustände unseres Modells die folgenden Eigenschaften:

- Es gibt für jede Auftragsphase I, I=1,..,L, eine schnelle, **annähernd rekurrente** Zustandsmenge C_I. Dies sind die Zustände der Phase I, in denen keine Störung aufgetreten ist. Der Wechsel zwischen zwei Phasen geschieht dann nach Abschnitt 2.1 mit vergleichsweise sehr kleinen Raten, so daß die NCD-Eigenschaft erfüllt ist, vgl. (2.1-7), (2.1-8).

- Es gibt eine schnelle, **transiente** Zustandsmenge C_T, wobei T=L+1 ist. Diese Menge ist mit den rekurrenten Zustandsmengen C_I nur lose gekoppelt, weil die Auftritts-Raten für Störungen sehr klein sind. Weil die Behandlung von Speicherfehlern mit vergleichsweise sehr hohen Raten geschieht, sind die Zustände in C_T nur transient. D.h. die stationäre Zustandswahrscheinlichkeit für C_T ist ebenfalls sehr klein.

Modelliert man den Wechsel zwischen den Zuständen durch einen Markovprozeß, so können die Zustandsübergänge durch die Übergangsmatrix in Abbildung 3.1 verdeutlicht werden.

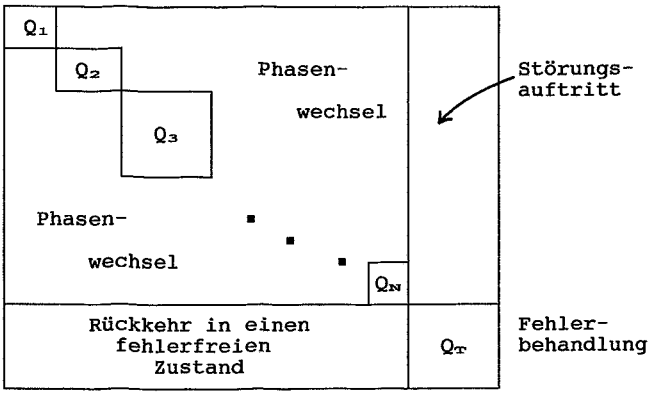


Abb. 3.1: Zustandsübergangsmatrix eines Auftrags

Die quadratische Submatrix, die nur die fehlerfreien Zustände berücksichtigt, d.h. die mit dem Leistungsmodell korrespondiert, besitzt annähernd Blockdiagonalform. Die Raten außerhalb der Teilmatrizen Q_I, I=1,..,L, entsprechen einem Phasenwechsel und sind vergleichsweise sehr klein (siehe oben). Also gilt hier die NCD-Voraussetzung

von Courtois /COUR 77/. Diese Modellsicht entspricht derjenigen aus Abschnitt 2.1, vgl. (2.1-7) und (2.1-8).

Die Übergangsraten von den Q_I, I=1,..,L, zur schnellen transienten Zustandsmenge Q_T erfolgt mit den sehr kleinen, phasenspezifischen Auftritts-Raten von (nicht maskierten) Störungen. Die gesamte Matrix besitzt aber nicht die NCD-Form, weil Q_T mit den relativ hohen Raten für eine Fehlerbehandlung verlassen wird.

Dies verdeutlicht auch noch einmal, daß Q_T einer schnellen, transienten Zustandsmenge entspricht. Um diese Zustandsmenge zu betrachten, ist eine **transiente Analyse** /BOTR 86/, /ROWI 88/ erforderlich. Wie bereits erwähnt, ergibt sich bei stationären Betrachtungen für Q_T eine Verweildauer von nahezu 0. Zeitabhängige Modell-Größen werden werden in Abschnitt 7.3 (Auftrags-spezifische Analyse) bestimmt.

3.2 Wechselwirkungen zwischen den Teilmodellen

Wir wollen nun auf die Wechselwirkungen eingehen, die zwischen den drei oben genannten Teilmodellen bestehen. Diese können, wie in der folgenden Abbildung veranschaulicht, miteinander verknüpft werden.

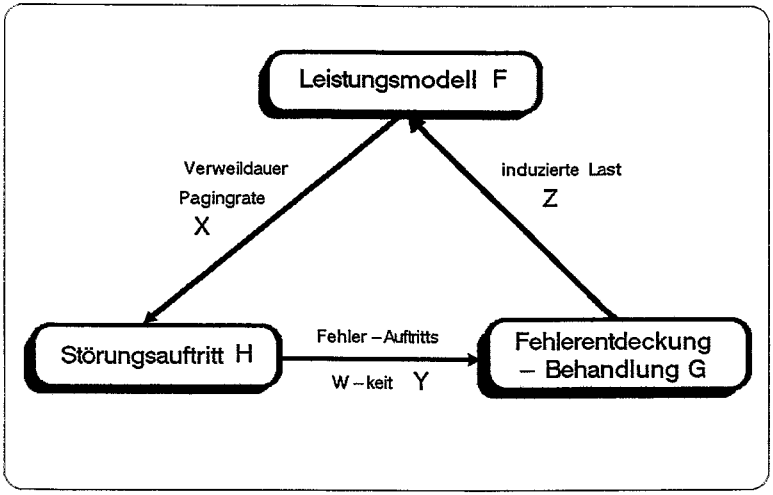


Abb. 3.2: Zusammenwirken der drei Teilmodelle

Die durch Abbildung 3.2 beschriebenen Wechselwirkungen zwischen den Teilmodellen lassen sich formal durch ein System **nichtlinearer Gleichungen** /ORRH 70/ beschreiben. Dazu wollen wir zunächst einige Parameter einführen.

Jedes Teilmodell liefert spezielle Ergebnisgrößen. Diese Größen gehen wiederum als Eingabe-Parameter in ein anderes Teilmodell ein. Seien mit den Vektoren **X** Ergebnisgrößen des Leistungsmodells, mit **Y** Ergebnisse des Modells zum Auftreten von Störungen, und mit **Z** solche des Modells zur Fehlerentdeckung und -behandlung bezeichnet. Dann bestehen **X, Y** und **Z** im einzelnen aus den folgenden Größen.

Der dem Leistungsmodell zugeordnete Ergebnisvektor $\mathbf{X}=(X_1,..X_p)$ enthält u.a. die folgenden Leistungsmaße:

- die Verweildauer V eines Auftrags,
- die Paging-Rate Φ^P des Auftrags,
- die working-set Größe w(T) des Auftrags,
- die Zugriffsraten β_i auf einzelne Speicherseiten s_i.

Der Ergebnisvektor $\mathbf{Y}=(Y_1,..,Y_q)$ des Störungsmodells enthält u.a.:

- die Wahrscheinlichkeit q_i mit der bei einer virtuelle Seite s_i ein Fehler entdeckt wird. Dabei werden verschiedene Störungs-Arten unterschieden.

Der Vektor $\mathbf{Z}=(Z_1,..,Z_r)$ des Modells zur Fehlerentdeckung und -behandlung quantifiziert die durch Fehler induzierte Last, d.h. speziell:

- die Anzahl N^F der entdeckten Fehler,
- die durch Fehler verlängerte Bearbeitungsdauer B^R eines Auftrags,
- die Anzahl Γ^R der zur Fehlerbehandlung erforderlichen Seiteneinlagerungen.

Mit Hilfe dieser Vektoren lassen sich für unser Modell die folgenden Gleichungen aufstellen:

$$\mathbf{X} = F(\mathbf{Z}) \ , \qquad (3.2\text{-}1)$$

d.h. die durch Fehler induzierte Last geht ins **Leistungsmodell F** zur Berechnung von Leistungsindizes, wie z.B. der Verweildauer, ein.

$$\mathbf{Z} = G(\mathbf{Y}) \ , \qquad (3.2\text{-}2)$$

d.h. ins **Modell G der Fehlerentdeckung und -behandlung** gehen die Auftritts-Wahrscheinlichkeiten von Störungen ein. Als Ergebnis dieses Modells erhält man die durch Fehler induzierten Last.

$$\mathbf{Y} = H(\mathbf{X}) \ , \qquad (3.2\text{-}3)$$

d.h. im **Modell H zum Störungs-Auftritt** werden Leistungsindizes berücksichtigt.

Die Gleichungen (3.2-1) bis (3.2-4) lassen sich für die Leistungsuntersuchung zu

$$\mathbf{X} = F(G(H(\mathbf{X}))) \; . \qquad (3.2\text{-}4)$$

zusammenfassen. Die Lösung von (3.2-4) ist der Fixpunkt dieser Gleichung. Der als Lösung dieser Fixpunkt-Gleichung bestimmte Ergebnisvektor **X** liefert Leistungsindizes, die die Wechselwirkungen mit den aufgetretenen Fehlern berücksichtigen.

Entsprechende Fixpunkt-Gleichungen sind natürlich auch für die Parameter-Mengen **Y** und **Z** möglich.

Lösung nichtlinearer Gleichungen

In einigen Untersuchungen zur Leistungsbewertung werden nichtlineare Gleichungen benutzt /BARD 78/, /LAZO 84/, /SCHO 86/. Einen Überblick bietet /SSLM 84/. Zur Lösung solcher Gleichungen werden nach Vorgabe von Startwerten sukzessive neue Funktionswerte berechnet, bis dieses **iterative Verfahren** /SCHW 84/ konvergiert.

Im allgemeinen ist es jedoch sehr schwierig, die Existenz und Eindeutigkeit sowie die Konvergenz eines solchen Verfahrens theoretisch nachzuweisen. In /ORRH 70/ werden die Bedingungen für die Existenz von Lösungen nichtlinearer Gleichungen untersucht. Den Nachweis für die Existenz eines Fixpunktes liefert bspw. das sogenannte Brouwer Fixpunkt-Theorem. Genauere Untersuchung dazu werden ebenfalls im Kapitel 7 (Integration der Teilmodelle) durchgeführt.
Zunächst sollen jedoch in den folgenden Kapiteln die drei genannten Teilmodelle entwickelt werden.

4 Auftreten von Störungen

In diesem Kapitel untersuchen wir das Auftreten von Störungen, die sehr unterschiedliche Ursachen haben können. Es gibt Störungen, die im weitesten Sinne **ereignisunabhängig** sind, d.h. deren Auftreten nicht mit bestimmten Ereignissen im Speicher verbunden ist, sondern vielmehr als zufällig oder spontan erscheint. Ein Modell für das Auftreten solcher Störungen soll im Abschnitt 4.1 abgeleitet werden. Dabei wird insbesondere das Ausmaß einer einzelnen Störung im Speicher betrachtet, d.h. wir untersuchen, wieviele Seiten von einer bestimmten Störung betroffen sind.

Bestimmte Störungs-Arten können jedoch ausschließlich in Verbindung mit speziellen Ereignissen auftreten. Im Abschnitt 4.2 werden diese **ereignisabhängigen** Störungen betrachtet. Dazu untersuchen wir speziell Störungen **beim Seitenzugriff** und **beim Ein-/Auslagern** einer Seite.

Alle genannten Störungsursachen sind **lastinduziert**, d.h. sie hängen von bestimmten, jeweils unterschiedlichen Lastmerkmalen ab.
Bei einer realistischen Modell-Entwicklung müssen deshalb die relevanten Last-Charakteristika ins Störungs-Modell einfließen. In diesem Kapitel wollen wir dies in den verschiedenen Modellen für das Auftreten von Störungen im Speicher berücksichtigen.

4.1 Ereignisunabhängige Störungen

Zunächst soll ein stochastisches Modell angegeben werden, mit dem sich die Anzahl aufgetretener ereignisunabhängiger Störungen bestimmen läßt. Weil diese Störungen sehr unterschiedliche Auswirkungen aufweisen können, wird anschließend untersucht, in welchem Ausmaß der Speicher von solch einer Störung betroffen sein kann.

4.1.1 Auftreten ereignisunabhängiger Störungen

In diesem Abschnitt wollen wir ereignisunabhängige Störungen des Hauptspeichers betrachten. Diese Störungen werden nicht durch ein bestimmtes Ereignis, wie z.B. einen Zugriff oder einen Seitenwechsel

induziert, vielmehr scheint ihr Auftreten, bei isolierter Betrachtung des Speicherverwaltungs-Systems, "spontan" oder zufällig zu sein. Typische Beispiele für die Ursachen ereignisunabhängiger Störungen sind /BAEH 88/, /BOME 86/:

- Störungen durch α-Partikel, sogenannte **soft errors**, die durch atomare Prozesse ausgelöst werden, und von System-Aktivitäten im Speicher oder auch von thermischen Effekten unabhängig sind. Diese Störungs-Art macht den größten Teil von Störungen innerhalb des Speichers aus /BOHS 80/. Die durch α-Partikel verursachte Störungs-Rate steigt mit der Integrationsdichte der Speicherchips /OHM 79/, /GEIL 79/.

- Kosmische Strahlung /MAWO 79/.

- Durch Fehler in der Spannungsversorgung induzierte Speicherfehler. Diese können im weitesten Sinne ebenfalls als ereignisunabhängig angenommen werden, weil sie von keinem Ereignis in der Speicherverwaltung direkt abhängen. Auch Störungen in der Spannungsversorgung üben einen entscheidenden Einfluß auf die Zuverlässigkeit von Speichern aus /ELSI 80/.

In dieser Arbeit werden alle ereignisunabhängigen Störungen ausschließlich im folgenden Sinne als **lastinduziert** aufgefaßt: Die Wahrscheinlichkeit, daß ein bestimmter Auftrag von einem Speicherfehler betroffen wird, hängt von

- der Anzahl der belegten Speicherseiten und

- der Verweildauer dieser Seiten im Speicher

ab.

Sollen Störungen modelliert werden, die mit keinem bestimmten Ereignis korrelieren, dann ist es realistisch, vereinfachend anzunehmen, daß die zeitlichen Abstände zwischen zwei Störungen identisch verteilt und stochastisch unabhängig sind. D.h. wir können das Auftreten von ereignisunabhängigen Störungen durch einen **Erneuerungsprozeß** /COX 62/, /SCHN 83/ modellieren. Formal läßt sich dann definieren:

- Es gibt für jede ereignisunabhängig auftretende Störungs-Art einen allgemein verteilten Störungsprozeß, der mit der Rate δ_a den Hauptpeicher stört. Dieser Störungsprozeß soll die Eigenschaften eines Erneuerungsprozesses aufweisen, d.h. jede Störung des Hauptspeichers ist ein Erneuerungspunkt, vgl. Abbildung 4.1.

Von einer einzelnen Störung können eine oder auch mehrere Speicherseiten betroffen sein. Die Auswirkungen einzelner Störungen werden in Abschnitt 4.1.2 untersucht.

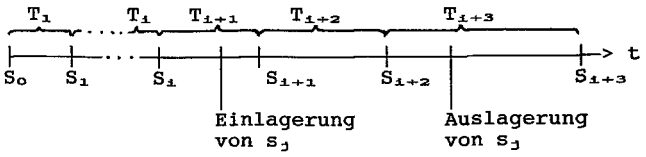


Abb. 4.1: Erneuerungsprozeß zur Modellierung ereignisunabhängiger Störungen im Hauptspeicher.

Die Zeiten T_i zwischen zwei aufeinanderfolgenden Störungen S_{i-1} und S_i sind also für alle i=1,..,n voneinander unabhängig und gleich verteilt. D.h.

$$F_S(t) := \text{Prob}[S_i - S_{i-1} \leq t] \quad , \quad \text{für alle } i=1,2,... \qquad (4.1\text{-}1)$$

bezeichnet die Verteilungsfunktion der Störungsabstände. $f_S(t)$ ist die zugehörige Verteilungsdichte. Der i-te Erneuerungspunkt ist gegeben durch:

$$S_i = \sum_{1 \leq j \leq i} T_j \quad . \qquad (4.1\text{-}2)$$

Die Verteilungsfunktion von S_i sei $F_{Si}(t)$. Dann ist nach /SCHN 83/ (mit dem Faltungs-Operator *) die zugehörige Dichte:

$f_{Si}(t) = f_S^{*i}(t)$ mit

$f_S^{*i}(t) := f_S^{*i-1} * f_S(t)$,

wobei $f_S^{*1} = f_S(t)$ und $f^{*0} * g = g$. (4.1-3)

Für die Verteilungsfunktion $F_{Si}(t) := \text{Prob}[S_i \leq t]$ ergibt sich:

$$F_{Si}(t) = f_S^{*(i-1)} * F_S(t) \quad . \qquad (4.1\text{-}4)$$

Im allgemeinen interessiert uns nicht, wieviele Störungen in der gesamten bisherigen Lebensdauer, d.h. im Intervall [0,t], aufgetreten sind. Vielmehr betrachten wir meist eine einzelne Seite, und wollen wissen, wieviele Störungen bei dieser Seite während ihrer Verweildauer im Speicher stattgefunden haben. D.h. der Beginn des von uns betrachteten Zeitraums fällt nicht mit dem ersten Erneuerungspunkt S_0 zusammen, sondern liegt zufällig zwischen zwei benachbarten Erneue-

rungspunkten S_i und S_{i+1}. In diesem Fall besitzt der Abstand zum ersten Erneurungspunkt S_{i+1} nicht mehr dieselbe Verteilung wie die übrigen Störungsabstände. Es liegt also ein **allgemeiner (verzögerter) Erneuerungsprozeß** vor, und die Zeit zum ersten Erneuerungspunkt im Betrachtungszeitraum ist die **Vorwärtsrekurrenzzeit** S_{VR} des Störungsabstandes T.
Deren Verteilung ist im stationären Fall asymptotisch

$$F_{SVR}(t) = \delta_S \cdot \int_0^t [1-F(t')]dt' \quad , \tag{4.1-5}$$

mit der mittleren Störungsrate $\delta_S=E[T]^{-1}$ /GASC 84/. Die zugehörige Verteilungsdichte ist:

$$f_{SVR}(t) = \delta_S \cdot [1-F(t)] \quad . \tag{4.1-6}$$

Bei einem verzögerten Erneuerungsprozeß sind die Nullpunktabstände S_i der Erneuerungspunkte wie folgt verteilt:

$$F^V_{Si}(t) = \begin{cases} f_{SVR}(t) * f_S^{*(i-2)} * F_S(t) & , \text{falls } i \geq 2 \\ F_{SVR}(t) & , \text{falls } i=1 \end{cases} \tag{4.1-7}$$

Für die Verteilungsdichte gilt dann:

$$f^V_{Si}(t) = f_{SVR} * f_S^{*i-1}(t) \quad . \tag{4.1-8}$$

Es sei $N^U(t)$ die Anzahl der Erneuerungspunkte in einem Intervall $[t_o,t_o+t]$ und $Pr^U(i,t)$ die Wahrscheinlichkeit für genau i Erneuerungspunkte im Intervall $[t_o,t_o+t]$. Dann ist:

$$Pr^U(i,t) := Prob[N^U(t)=i]$$

$$= F^V_{Si}(t) - F^V_{Si+1}(t) \quad , \tag{4.1-9}$$

und die Wahrscheinlichkeit, daß es wenigstens i Erneuerungspunkte in $[t_o,t_o+t]$ gibt, ist:

$$Prob[N^U(t)\geq i] = F^V_{Si}(t) \quad . \tag{4.1-10}$$

Die mittlere Anzahl aller Erneuerungspunkte N^U während einer Zeitspanne der Länge t berechnet sich nach /GASC 84/ mit Hilfe des mittleren Erneuerungsabstandes $\bar{s}$, d.h. der mittleren Zeitdauer zwischen zwei Erneuerungszeitpunkten. Er ist gegeben durch:

$$\bar{s} = \int_0^\infty [1 - F_S(t)]dt \quad . \tag{4.1-11}$$

Sonderfall: Exponential-Verteilung

Selten vorkommende Ereignisse, wie z.B. Störungen, werden oft durch einen Poisson-Prozeß modelliert. Dann sind die zeitlichen Abstände zweier Störungen mit der Rate δ_S exponential verteilt, d.h.

$$F_S(t) = 1 - e^{-\delta_S t} \quad . \tag{4.1-12}$$

Der mittlere Erneuerungsabstand ist in diesem Fall durch

$$\bar{s} = 1/\delta_S \tag{4.1-13}$$

gegeben, und die Anzahl der aufgetretenden Störungen ist durch die Poisson-Verteilung bestimmt:

$$Pr^U(k,t) := Prob[N^U(t) = k] = \frac{(\delta_S \cdot t)^k}{k!} \cdot e^{-\delta_S t} \quad . \tag{4.1-14}$$

Wegen der Gedächtnislosigkeit der Exponentialverteilung muß in diesem Sonderfall nicht zwischen gewöhnlichen und verzögerten Erneuerungsprozessen unterschieden werden, denn die Vorwärtsrekurrenzzeit einer Exponentialverteilung ist wiederum exponential mit derselben Rate verteilt.
Je nach angewandtem Verfahren der Fehlerentdeckung wird zu unterschiedlichen Zeitpunkten überprüft, ob Störungen aufgetreten sind. Bei allen Verfahren hängt die Wahrscheinlichkeit, mit der eine Seite gestört wurde, offensichtlich davon ab, wie lange sie seit der letzten Fehlerüberprüfung im Speicher eingelagert war.

Mit Hilfe von $\bar{s}$ kann nun die mittlere Anzahl der Störungen $\bar{N}^U$, die während der Verweildauer V_i einer Seite s_i im Hauptspeicher aufgetreten sind, berechnet werden:

$$\bar{N}^U = V_i/\bar{s} \quad . \tag{4.1-15}$$

Die Gleichung (4.1-9), bzw. im Exponentialfall die Gleichung (4.1-14), geben die Verteilung der Anzahl der im Zeitraum t aufgetretenen Störungen an.

4.1.2 Auswirkungen ereignisunabhängiger Störungen

Die im vorhergehenden Abschnitt 4.1.1 beschriebenen ereignisunabhängigen Störungen haben sehr unterschiedliche Ursachen. Offen blieb bisher jedoch die Frage nach den Auswirkungen einer einzelnen Störung. Für unser betriebssystem-orientiertes Modell eines seitenverwalteten Speichers muß festgelegt werden, wieviele Seiten von einer bestimmten Störung betroffen sein können. Diese Anzahl hängt in starkem Maße von der Störungsursache ab. Bspw. kann bei einer durch α-Partikel induzierten Störung davon ausgegangen werden, daß nur eine einzige Seite betroffen ist. Fehler in der Spannungsversorgung führen aber u.U. zu Verfälschungen aller Seiten, deren Chips sich auf der betroffenen Platine befinden. Im folgenden wird ein genaueres Modell für das (physikalische) Ausmaß von Speicherfehlern festgelegt. Dazu wollen wir zunächst die Speicher-Architektur und ihre Realisierung durch die Hardware betrachten.

Physikalische Struktur von Halbleiterspeichern

Die physikalische Struktur von Halbleiterspeichern läßt sich durch die Anordnung und Organisation von Speicher-Bausteinen beschreiben. Ein weiteres Kriterium ist die logische Struktur des Speichers, d.h. wie Speicherseiten im physikalischen Speicher abgelegt werden.
Ein Halbleiterspeicher ist nach /RUTL 85/, /CHRU 84/ aus den folgenden Bestandteilen hierarchisch aufgebaut:

- Der Speicher besteht aus E identischen Speicherkarten, die in D sogenannte **bit-planes** unterteilt sind.
- Jedes bit-plane enthält eine Matrix von A·B **Speicher-Chips**.
- Jeder Speicher-Chip umfaßt wiederum X·Y Ein-Bit-**Speicherzellen**, die jeweils ein Bit realisieren.

Abbildung 4.2 zeigt die Struktur eines 2 Mbyte großen aus 16 Kbit Chips aufgebauten Speichers. Innerhalb dieser hierarchischen Anordnung von Speicher-Bausteinen wird ein Speicherwort im allgemeinen so realisiert, daß es aus E·D Bits besteht, wobei jedes Bit aus einer anderen bit-plane stammt. Für ein spezielles Bit im Speicherwort ist die Adresse des entsprechenden Speicher-Chips, sowie der Speicherzelle innerhalb dieses Chips bei jeder bit-plane identisch. In unserem Beispiel besteht ein Speicherwort aus 18·4=72 Bits.

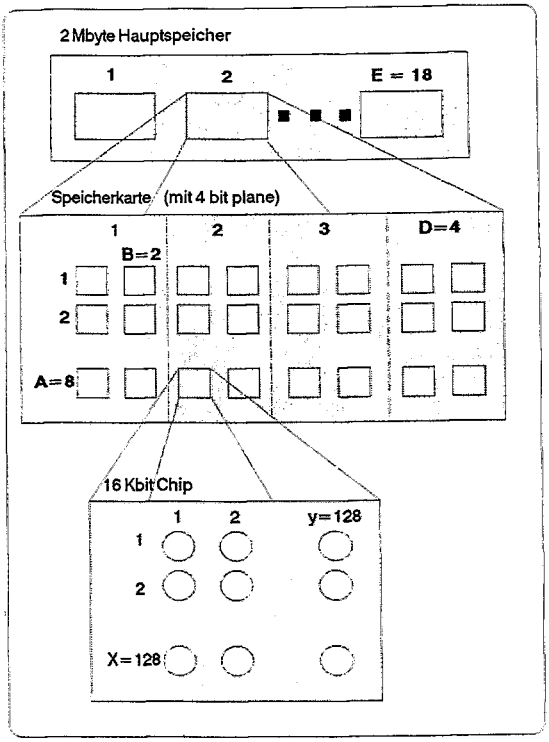


Abb. 4.2: **Organisation eines 2 Mbyte Hauptspeichers mit 16 Kbit Speicher-Chips**

Weil jedes Bit eines Speicherwortes auf einem anderen Chip abgespeichert ist, spricht man auch von **bit-per-chip-Organisation.**

Bei der oben beschriebenen Organisationsform gibt es im System E·D verschiedene bit-planes. Dann ist

$$W = A \cdot B \cdot X \cdot Y \qquad (4.1\text{-}16)$$

die **Anzahl der Speicherworte im Hauptspeicher.**

Veränderungen in der Speicher-Architektur durch Hochintegration

Mit fortschreitender Technologie können immer mehr Bits auf einem Speicher-Baustein realisiert werden. In den beiden oben zitierten Arbeiten wurden z.B. Speicher-Chips mit einer Kapazität von 16 Kbits zugrunde gelegt. Inzwischen sind diese aber vollständig von 64 Kbit oder 256 Kbit Chips verdrängt worden; auf dem Markt gibt es mittlerweile sogar 1Mbit Speicher-Chips. Aufgrund dieser Entwicklung ist es möglich, bereits auf einer Platine sehr große Halbleiterspeicher zu realisieren. Deshalb gibt es oft auch nur eine Speicherkarte, bei der jede Chip-Reihe eine bit-plane realisiert. D.h. in unsererer Notation gilt E=1, A=1 und D ist die Anzahl der Chip-Reihen auf der Speicherkarte.

Beispiel

Als Beispiel für die Speicherorganisation auf einer Karte zeigt die folgende Abbildung 4.3 eine 1-Mbyte-Speicherkarte, wie sie in VAX-Rechnern benutzt wird /BLGM 88/.
Bei dieser Realisierung hat jedes Speicherwort eine Länge von k=32 Datenbits und wird durch r=7 Prüfbits mit einem **SEC-DED** Code (**single-error correcting, double-error detecting code**) /CHHS 84/ gesichert, d.h. einzelne Bit-Fehler innerhalb eines Speicherwortes können korrigiert, alle zweifachen Bit-Fehler erkannt werden.

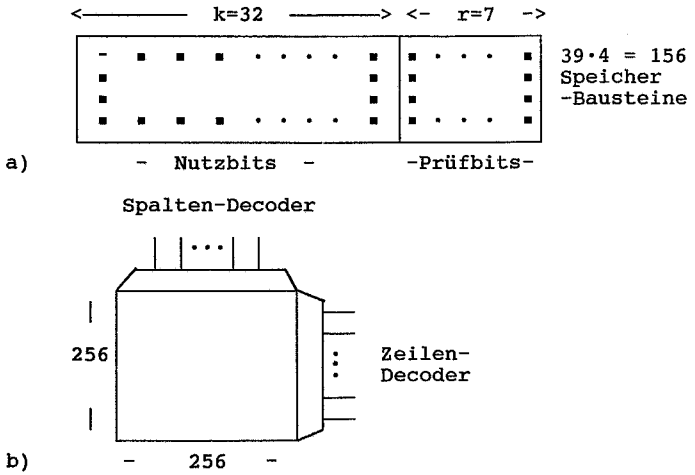


Abb. 4.3: a) 1-Mbyte-Speicherkarte eines VAX-Rechners
b) 64-Kbit-Speicherchip

Maßnahmen zur Erhöhung der Zuverlässigkeit in Speichern

- **Coderedundanz**: Mit den immer größeren Halbleiterspeichern wächst auch die Zahl der Speicherfehler. Aus diesem Grund sind fast alle großen Speicher durch fehler-korrigierende Codes geschützt. Normalerweise werden, wie im Beispiel oben, SEC-DED-Codes verwendet, die pro Wort ein verfälschtes Bit korrigieren und zwei falsche Bits erkennen können.

Werden in Speichern **permanente Fehler** (hard oder solid failures) erkannt, so sind folgende Maßnahmen möglich /AICH 84/:

- **page deallocation**: betroffene Speicherseiten werden logisch entfernt, d.h. sie werden nicht mehr benutzt.

- **card replacement**: Speicherkarten, bei denen Chips ausgefallen sind, werden nach verschiedenen Wartungsstrategien ausgetauscht /BOHS 80/.

- **spare switching**: im Fehlerfall werden defekte Speicher-Chips direkt durch einen Ersatz-Chip ohne manuelles Eingreifen ausgetauscht.

Modelle physikalischer Fehler

In der Literatur wurden in den letzten Jahren eine Reihe von Fehlern betrachtet, die anhand empirischer Untersuchungen in Halbleiterspeichern nachgewiesen worden sind /CHRU 84/, /LIHA 84/, /RUTL 85/, /BLGM 88/. Dabei werden die folgenden Fehler-Arten unterschieden:

- Fehler innerhalb eines Chips:
  - Ausfall einer Speicherzelle (single cell),
  - Ausfall einer Zeile (cell row),
  - Ausfall einer Spalte (cell column),
  - Ausfall des gesamten Speicher-Chips (chip kill).

- Fehler innerhalb einer bit-plane:
  - Ausfall einer Chip-Zeile (chip row),
  - Ausfall einer Chip-Spalte (chip column),
  - Ausfall einer bit-plane (entire bit-plane),

- Ausfall einer Speicherkarte (entire card).

Folgende Gründe werden in /BLGM 88/ für den Ausfall von Spalten und Zeilen innerhalb eines Chips, bzw. auf einer Platine genannt:

- Fehler/Ausfall des Spalten- bzw. Zeilen-Decoders eines Chips.

- Fehler des Spalten bzw. Zeilen-Treibers einer Speicherkarte.

Bei SEC-DED Codes wird in jedem Speicherwort ein falsches Bit toleriert, so daß ein nicht korrigierbarer Fehler (**UE-uncorrectable error**) auftritt, wenn zwei Bits innerhalb eines Speicherwortes verfälscht sind. D.h. bei einer bit-per-chip Organisationsform kann sogar der Total-Ausfall eines Chips toleriert werden.
In der Praxis ereignen sich UE's oft dadurch, daß sich im Laufe der Betriebsdauer des Systems permanente Speicherfehler akkumulieren. Dann können die um einige Größenordnungen häufigeren, transienten Fehler /BOHS 80/, einen UE verursachen, auch wenn sie nur eine Speicherzelle betreffen.
In der Literatur werden verschiedene Wartungs-Strategien genannt, um diesen Effekt zu verhindern /BOHS 80/. Diese basieren zum einen auf dem (präventiven) Austausch von Chips bzw. Speicherkarten, wenn sie - u.U. nur teilweise - von permanenten Fehlern betroffen sind. Eine ebenfalls häufig benutzte und durch Software realisierbare Strategie ist es, einzelne Speicherbereiche logisch auszugrenzen (page deallocation).

Unter der Voraussetzung der genannten Fehlermodelle wurde mit Hilfe spezieller Simulationsprogramme /CHRU 84/, /LIHA 84/ untersucht, wie groß die Rate der nicht korrigierbaren Hauptspeicherfehler ist.

Probleme der Fehlererkennung/-korrektur bei Hochintegration

Seit einigen Jahren sind bereits Speicherchips mit einer Kapazität von 1 Mbit verfügbar. Durch diese enorme Erhöhung der Bit-Dichte ergeben sich verschiedene Effekte, die das Problem des Auftretens und der Behandlung von Speicherfehlern verschärfen.

- Mit steigender Bit-Dichte wächst die Zahl der durch α-Partikel induzierten Fehler /BOHS 80/, /BLGM 88/.

- Es wird möglich, daß ein einziges α-Partikel mehrere Speicherzellen verfälscht /LIHA 84/.

- Bei extrem hoher Kapazität eines Speicherchips, wird, um die Anzahl der Speicher-Bausteine zu minimieren, oft auf die bit-per-chip Organisation verzichtet. In diesem Fall wird eine

multiple-bit-per-chip Organisationsform gewählt /AICH 84/, /BOME 86/.

Daraus folgt, daß zum einen die Speicherfehler-Rate beträchtlich wächst, zum anderen sich die Anzahl der durch einen Fehler betroffenen Bits erhöht. So verfälscht bei Höchstintegration der Totalausfall eines Chips im Vergleich zu Speicher-Chips älterer Technologie sogleich ein Vielfaches an Speicherzellen (1Mbit). Wird eine multiple-bit-per-chip-Organisation realisiert, so kann auch nicht mehr angenommen werden, daß Bit-Fehler innerhalb eines Speicherwortes stochastisch unabhängig sind. Insbesondere Mehr-Bit-Fehler treten mit weitaus größerer Wahrscheinlichkeit auf. Dabei übersteigen gerade Mehr-Bit-Fehler die Fähigkeiten von SEC-DED-Codes, denn sie können nicht mehr maskiert werden, vielmehr führen sie zwangsläufig zu einem nicht durch die Codes korrigierbaren Fehler.

In unserer betriebssystem-orientierten Modellsicht interessiert uns nicht nur die Anzahl der von einer Störung betroffenen Speicherworte, sondern vor allen Dingen die Anzahl der Seiten, die von einem einzigen Fehler betroffen werden. Dazu müssen wir zunächst die logische Struktur eines seitenorientierten Halbleiterspeichers betrachten.

Logische Struktur von Halbleiterspeichern

Die logische Struktur des Speichers gibt vor, wie sich sämtliche Worte einer Seite auf die Speicherchips verteilen. Wir benötigen diese Struktur für unser Modell, um abzuschätzen, wie sich physikalische Fehler der "Chip-Ebene" auf die "logische Seiten-Ebene" auswirken.
Der Hauptspeicher ist fest in Seiten, die oft auch **Kacheln** (frames) genannt werden, aufgeteilt. Dabei sind die Speicherworte einer Seite durch einen **konsekutiven Adreßbereich** bestimmt.

Nehmen wir an, daß der Hauptspeicher aus H Seiten besteht, die jeweils W_S Speicherworte enthalten. Bei einer **bit-per-chip-Organisation** liegen die jeweils i-ten Bits aller W_S Speicherworte einer Seite in W_S konsekutiven Speicherzellen eines Chips der i-ten Reihe.
Es gilt offensichtlich für die **Anzahl W_S der Speicherworte pro Seite**

$$W_S = W/H \ , \qquad (4.1\text{-}17)$$

mit W nach (4.1-16).
Bei einer bit-per-chip Organisation enthält jeder Speicher-Chip einzelne Bits von X·Y verschiedenen Speicherworten, d.h. die **Anzahl S_C der Seiten, die in einem Chip** enthalten sind, beträgt:

$$S_C = X \cdot Y / W_S \ . \qquad (4.1\text{-}18)$$

Dies ist auch die Anzahl der Seiten, die von einem einzelnen Chip-Ausfall betroffen sind.

In der Fortsetzung des oben angegebenen Beispiels wollen wir nun für unser Modell der physikalischen Speicherfehler die Anzahl der betroffenen Seiten bestimmen.

Beispiel: (Fortsetzung)

Betrachten wir die 1Mbyte Speicherkarte des Beispiels nach Abbildung 4.2. Insgesamt sind

$W = 4 \cdot 256 \cdot 256 = 2^{18} = 262.144$ 32-Bit-Speicherworte

im Speicher enthalten. Es sei angenommen, daß jede Seite W_S=1024 Worte, d.h. 4 Kbyte beinhalte. Daraus folgt, daß der Speicher insgesamt $H = 2^8 = 256$ Seiten enthält.

Jeder Speicher-Chip enthält also einzelne Bits aus den Worten von

$S_C = 2^{16}/2^{10} = 2^6 = 64$ Seiten.

D.h. es werden bei unseren 64 Kbit Speicher-Chips von den 256 Zeilen (word-lines) jeweils 4 benötigt, um eine Speicherseite zu realisieren. Betrachtet man hingegen eine Spalte (bit-line) eines Chips, so enthält sie Bits sämtlicher 64 Seiten des Speicherchips.
Damit kann die Wirkung physikalischer Speicherfehler auf Speicherseiten durch die folgende Tabelle 4.3 beschrieben werden.

Bei der Tabelle 4.3 muß beachtet werden, daß die genannten Auswirkungen nicht ohne weiteres Aussagen über die Anzahl der nicht korrigierbaren Fehler (UE's) liefern. Bei fast allen genannten Fehler-Arten wird - bit-per-chip Organisation vorausgesetzt - nur ein einzelnes Bit in u.U. sehr vielen Speicherworten verändert. D.h. diese Fehler können, falls keine weiteren (permanent) falschen Bits in den betroffenen Speicherworten vorhanden sind, durch SEC-DED-Codes korrigiert werden. Lediglich bei den Fehler-Arten "chip-row" und "entire card" ist es sicher, daß mehrere Bits pro Wort verändert werden, und somit gewiß, daß UE's auftreten. Die genauen Raten der UE's sind in den anderen Fällen kaum analytisch zu bestimmen, weshalb man sich der erwähnten Simulatoren /CHRU 84/, /LIHA 84/ bedient, um festzustellen, wie häufig zwei Bits innerhalb eines Wortes betroffen sind.
Wenn jedoch bei hoher Integrationsdichte eine multiple-bit-per-chip Organisation gegeben ist, führen fast alle Fehler-Arten zu einem UE.

Tabelle 4.1: Anzahl der von Speicher-Chip-Fehlern betroffenen Seiten

Fehler-Art	Anzahl der betroffenen Seiten
single cell	1 Seite
cell row	1 Seite
cell column	64 Seiten
chip kill	64 Seiten
chip row	64 Seiten
chip column	256 Seiten
entire card	256 Seiten

Zur vollständigen Spezifikation von Störungsprozessen muß nun formal angegeben werden, **wieviele** Seiten im Hauptspeicher von einer Störung betroffen sind. Die Auswirkung (**extension**) einer Störung kann folgendermaßen durch die Verteilung der Anzahl gestörter Seiten N^U spezifiziert werden:

$$e(k) := \text{Prob}[N^U = k] \quad , \quad \text{für } 1 \le k \le H \quad . \tag{4.1-19}$$

Die mittlere Anzahl der von einer Störung betroffenen Seiten ergibt sich dann durch

$$\bar{e} = \sum_{1 \le k \le H} k \cdot e(k) \quad . \tag{4.1-20}$$

Beispiel

Die Auftrittswahrscheinlichkeiten der genannten sieben Fehler-Arten seien folgendermaßen verteilt. Ein "single cell" Fehler soll mit der relativ großen Wahrscheinlichkeit 0.4 auftreten, weil er auch durch transiente Fehler verursacht wird. Für alle restlichen Fehler-Arten wird eine Auftrittswahrscheinlichkeit von 0.1 angenommen. Dann ergibt sich für die Verteilung der Anzahl gestörter Seiten (vgl. Tabelle 4.1):

$$e(k) = \begin{cases} 0.5 & \text{für } k = 1, \\ 0.3 & \text{für } k = 64, \\ 0.2 & \text{für } k = 256, \\ 0 & \text{sonst.} \end{cases}$$

Die mittlere Anzahl gestörter Seiten ist dann $\bar{e} = 70.9$.

4.2 Ereignisabhängige Störungen

Das Modell in Abschnitt 4.1 geht von ereignisunabhängigen Störungen aus, deren Auftreten nicht mit bestimmten Aktivitäten im Speicher verbunden ist. Sie erscheinen vielmehr zufällig, bzw. spontan und werden durch einen Erneuerungsprozeß modelliert.

Es gibt aber auch Störungen, die nur in Verbindung mit bestimmten Ereignissen im Speicher stattfinden können. Im weiteren werden wir insbesondere zwei Ereignisse berücksichtigen, die Fehler induzieren können.
Zunächst behandeln wir **Störungen**, die **beim Zugriff** auf ein Speicherwort einer Seite auftreten können; danach solche, die **beim Ein- und Auslagern** von Seiten entstehen.

4.2.1 Zugriffsinduzierte Störungen

In diesem Abschnitt schreiben wir nicht der Verweildauer einer Speicherseite ein **Störungs-Risiko** zu (siehe Abschnitt 4.1), sondern jedem einzelnen **Seitenzugriff**.
Dies ist sinnvoll, weil es bestimmte Störungen gibt, die ausschließlich beim Zugriff auf den Speicher auftreten können. Weiterhin gibt es andere Störungen, bei denen die Auftritts-Häufigkeit direkt proportional zur Seitenzugriffs-Rate ist.
Folgende Ursachen können für solche **zugriffsinduzierten** Störungen verantwortlich sein /BAEH 88/:

- Übersprechen auf Adreß- und Datenleitungen,
- Fehler beim Adreß-Dekodieren,
- durch Software verursachte Speicherfehler können offensichtlich nur bei einem Seitenzugriff erfolgen,
- die thermische Belastung hängt von der Zugriffshäufigkeit ab, denn bei großen Speichersystemen werden nur die selektierten Speicherchips mit der vollen Betriebsspannung versorgt (power down mode, stand by mode),
- eine Beeinflussung durch Nachbarzellen tritt nur auf, wenn auf diese Nachbarzellen zugegriffen wird,

- und letztendlich nimmt auch die Verweildauer im Arbeitsspeicher mit wachsender Zugriffswahrscheinlichkeit zu, sodaß auch ereignisunabhängige Störungen wahrscheinlicher werden. (In diesem Sinne lassen sich die ereignisunabhängigen Störungen nicht völlig getrennt von den ereignisabhängigen betrachten.)

Es gibt allerdings keinen offensichtlichen oder empirisch abgesicherten Zusammenhang zwischen Störungswahrscheinlichkeit und Zugriffshäufigkeit. Im weiteren gehen wir zur Modellierung zugriffsinduzierter Speicherfehler deshalb vom folgenden stochastischen Modell aus.

Modellannahme:

- Jede Störung beim Zugriff auf eine Seite wird als ein (stochastisch unabhängiges) **Bernoulli Experiment** aufgefaßt. D.h. es wird angenommen, daß mit der festen Wahrscheinlichkeit p_z bei einem Seitenzugriff eine Störung erfolgt.

Finden auf eine betrachtete virtuelle Seite während der Auftragsbearbeitung Z Zugriffe statt, dann ist demnach die Anzahl N^Z der aufgetretenden zugriffsinduzierten Störungen **binomisch verteilt**, d.h. es gilt:

$$Pr^Z(k,Z) := Prob[N^Z=k] = \binom{Z}{k} \cdot p_z^k (1-p_z)^{Z-k} \quad . \qquad (4.2\text{-}1)$$

Es besteht dann Proportionalität zwischen der Zugriffsanzahl und der Wahrscheinlichkeit, mit der eine Seite gestört wird. Sei X_j die Indikatorvariable eines gestörten Zugriffs, dann ist :

$$N^Z = \sum_{1 \le j \le Z} X_j$$

Für die mittlere Anzahl $\overline{N}^Z$ der durch Zugriffe induzierten Störungen ergibt sich dann offensichtlich (vgl. /SCHN 80/):

$$\overline{N}^Z = \sum_{1 \le j \le Z} Prob[X_j=1]$$

$$= Z \cdot p_z \quad . \qquad (4.2\text{-}2)$$

Dabei geht im letzten Schritt von (4.2-2) ein, daß die Wahrscheinlichkeit einer Störung für alle Zugriffe identisch (gleich p_Z) ist.

Wir gehen davon aus, daß zugriffsinduzierte Störungen immer nur eine einzige Seite betreffen, d.h. e(1)=1. Dies erscheint bei den oben genannten Fehlerursachen realistisch. Eine Erweiterung des

Modells auf beliebige andere Auswirkungen bereitet aber offensichtlich keine Probleme.

Im Modell werden lediglich Zugriffe auf Seiten betrachtet. Die Zugriffshäufigkeit bezieht sich dem **Detaillierungsgrad** unseres Modells entsprechend also ebenfalls auf Seiten. Es wird demnach nicht unterschieden, daß einzelne Wörter dieser Seite u.U. sehr verschiedene Zugriffswahrscheinlichkeiten aufweisen.

Jede entdeckte Verfälschung ergibt einen Fehler, der durch entsprechende Maßnahmen behoben werden kann. Die zu ergreifenden Maßnahmen sind meist von der Fehlerursache unabhängig, weil auf diese im allgemeinen nicht mehr zurückgeschlossen werden kann. Genauere Untersuchungen zu diesem Thema werden in Kapitel 5 vorgenommen. Dort wird u.a. auch Z_i, die Anzahl der Zugriffe auf eine Seite s_i bestimmt.

4.2.2 Seitenwechsel-induzierte Störungen

Ebenfalls ausgesproche kritische Ereignisse im Speicher sind die Seitenwechsel. Sowohl beim Einschreiben einer Seite in den Hauptspeicher, als auch bei der Übertragung auf den Hintergrundspeicher können Störungen auftreten.
Störungen auf den Übertragungskanälen und vor allen Dingen die relativ störanfällige Mechanik üblicher Plattenspeicher sorgen dafür, daß Seitenwechsel häufig zur Verfälschung von Speicherworten führen. Als mögliche Ursachen werden in /HSIT 87/, /BOME 86/ genannt:

- Fehler durch Staub bzw. Oxidation auf den Lese-/ Schreibköpfen der Platte,
- Übertragungsfehler,
- Fehler im Platten-Controller (z.B. falsche Positionierung des Lese/Schreib-Kopfes),
- Fehler bei der Umrechnung virtueller Adressen in physikalische.

Bei den vom System durchgeführten Verfahren zur Erkennung von Speicherfehlern kann normalerweise nicht unterschieden werden, ob die Störung beim eigentlichen Übertragungsvorgang oder bereits auf der Platte stattgefunden hat.

In diesem Abschnitt wollen wir ausschließlich Störungen betrachten, die bei der Übertragung stattfinden, d.h. durch einen Seitenwechsel induziert werden.

Eine Vorstellung über den Einfluß von Seitenwechseln auf die Zuverlässigkeit, gibt die Arbeit von Iyer et al. /IYBM 82/. Dort wird bei der Analyse empirischer Systemdaten eine sehr starke Korrelation zwischen der Anzahl von Systemausfällen und der Paging-Rate festgestellt.

Ähnlich wie in Abschnitt 4.2.1 jedem Seitenzugriff ein Fehlerrisiko zugeordnet wurde, kann dies somit auch für jeden Seitenwechsel geschehen.
Zur Modell-Bildung legen wir die folgenden Annahmen zugrunde:

Modellannahme:

- Bei jedem Seitenwechsel tritt mit Wahrscheinlichkeit p_P eine Störung auf. Haben insgesamt Γ Seitenwechsel stattgefunden, dann ist die Anzahl N^P, der dabei induzierten Störungen **binomisch verteilt**, d.h. es gilt analog zu (4.2-1):

$$Pr^P(k,\Gamma) := Prob[N^P=k] = \binom{\Gamma}{k} \cdot p_P^{\,k}(1-p_P)^{\Gamma-k}, \qquad (4.2\text{-}3)$$

mit dem Mittelwert

$$\bar{N}^P = \Gamma \cdot p_P \; . \qquad (4.2\text{-}4)$$

Die noch unbekannte Anzahl Γ der Seitenwechsel hängt von den Zugriffswahrscheinlichkeiten und der Fenstergröße ab. Sie wird im nächsten Kapitel berechnet. Dort werden auch verschiedene Methoden der Fehlerentdeckung vergestellt. Für die zur Fehlerbehandlung erforderlichen Maßnahmen ist bei allen Störungen der Zeitpunkt ihrer Entdeckung entscheidend . Auch dies ist Thema des folgenden Kapitels.

5 Entdeckung und Behandlung von Fehlern

In diesem Kapitel wollen wir mehrere Verfahren betrachten, mit deren Hilfe Speicherfehler entdeckt und behandelt werden. Dabei unterscheiden wir exemplarisch drei Vorgehensweisen.
Einmal können Speicherseiten **beim Ein- und Auslagern** überprüft werden, zum zweiten ist eine Überprüfung **bei jedem Zugriff** auf ein Speicherwort möglich, und schließlich können speziell bei fehlertoleranten Rechnern die Aufträge nach festen Ausführungsdauern durch sogenannte **Konsistenzprüfungen** getestet werden. Im nachfolgenden Abschnitt sollen diese verschiedenen Ansätze und die mit ihnen verbundenen Maßnahmen der Fehlerbehandlung genauer vorgestellt werden.

5.1 Verfahren der Fehlerentdeckung

Bei jedem Verfahren der Fehlerentdeckung liegt fest, wann Verfälschungen als Fehler bemerkt werden. Ein wichtiges Kriterium ist dabei die sogenannte **Latenzzeit** (error latency), d.h. die Zeit, die zwischen dem Auftreten einer Verfälschung und ihrer Entdeckung vergeht. Die Latenzzeit ist maßgebend für die erforderliche Fehlerbehandlung. Denn je später eine aufgetretene Verfälschung während der Auftragsbearbeitung bemerkt wird, um so mehr andere Speicherseiten können zwischenzeitlich durch fehlerhafte Daten verfälscht worden sein (Folgefehler / error propagation).
Liegt ein transienter Fehler vor, kann zur Fehlerbehandlung ein **Wiederanlauf** (**retry**) des gestörten Auftrags durchgeführt werden. Dann bestimmt die Latenzzeit u.a. auch den Ausführungs-Zeitpunkt, auf den ein gestörter Auftrag zurückgesetzt werden muß, um wieder in einen fehlerfreien, konsistenten Zustand zu gelangen /RAND 75/, /CLSK 87/. Zusätzlich müssen zur Wiedererlangung der **Datenintegrität** eventuell verfälschte Speicherseiten erneut eingelagert werden. Wir wollen im folgenden drei verschiedene Verfahren der Fehlerentdeckung unterscheiden, die zu jeweils anderen Latenzzeiten und damit auch unterschiedlichen Maßnahmen zur Fehlerbehandlung führen.

Modell 1: Fehlerentdeckung beim Ein-/Auslagern

- Die Überprüfung einer Seite erfolgt immer dann, wenn sie in den Hauptspeicher eingelagert oder von dort ausgelagert wird /HUMA 87/. Die eigentliche Überprüfung der Seite kann durch Coderedundanz (im einfachsten Fall durch Paritätsbildung) geschehen /DIGE 87/, /BAEH 88/ und ist in vielen Speicherverwaltungs-Systemen üblich. Bei diesem Verfahren ist die Latenzzeit direkt proportional zur mittleren Einlagerungsdauer einer Seite im Hauptspeicher, d.h. der Zeit, die eine Seite zwischen ihrer Ein- und Auslagerung im Speicher verbringt.

Fehlerbehandlung:

- Werden Fehler **beim Einlagern** einer Seite entdeckt, so sind diese nicht im Hauptspeicher, sondern beim Lesen des Hintergrundspeichers oder bei der eigentlichen Datenübertragung entstanden. Es handelt sich also nach Abschnitt 4.2 um Paging-Fehler, die durch die dort betrachteten ereignisabhängigen Störungen verursacht werden. Bei einem transienten Fehler reicht dann als Behandlungsmaßnahme ein erneutes Einlagern der referenzierten Seite. Wird **beim Auslagern** ein Seitenfehler entdeckt, so ist dieser während der Einlagerungsdauer im Hauptspeicher aufgetreten. Es sind dann zwei Aktionen zur erfolgreichen Fehlerbehandlung erforderlich: Zunächst muß die gestörte Seite erneut eingelagert werden. Des weiteren können alle Seiten, auf die nach der Einlagerung der gestörten Seite zugegriffen worden ist, durch den Fehler mit falschen Daten "infiziert" worden sein, so daß auch diese erneut eingelagert werden müssen. Nach der Wiederherstellung der Datenintegrität wird der betroffene Auftrag wiederholt, und zwar ab dem Zeitpunkt, zu dem die letzte Fehlerüberprüfung (also hier die Einlagerung der betroffenen Seite) durchgeführt wurde. Diese Form der Fehlerbehandlung erhöht nicht nur die Verweildauer, sondern auch die Ausführungsdauer des betroffenen Auftrags.

Modell 2: Fehlerentdeckung beim Speicherzugriff

- In diesem Modell wird vorausgesetzt, daß bei jedem Speicher**zugriff** eine Fehlerüberprüfung stattfindet. Dies kann dadurch gewährleistet werden, daß jedes Speicherwort durch fehlererkennende Codes (bspw. Hammingcodes /SCHN 83/) gesichert ist, und bei jedem Zugriff die Codezugehörigkeit geprüft wird. In diesem Fall kann eine Latenzzeit von nahezu Null angenommen werden. Bei fehlerkorrigierender Codes können einige Störungen sogar maskiert werden und führen erst gar nicht zu Fehlern /CHHS 84/.

Fehlerbehandlung:

- Wird ein Fehler schon beim Speicherzugriff entdeckt, so reicht im allgemeinen ein erneutes Einlagern der betroffenen Seite vom Hintergrundspeicher. Denn sobald ein verfälschtes Datum der betroffenen Seite benutzt wird, wird dieser Fehler sofort entdeckt, d.h. eine Ausbreitung des Fehlers auf andere Speicherseiten (**error propagation**) kann ausgeschlossen werden. Bei diesem Verfahren ist demnach kein Wiederanlauf (rollback) des betroffenen Auftrags erforderlich, und die Ausführungsdauer verlängert sich nicht nennenswert. Allerdings erhöht sich die Verweildauer um die für das Einlagern benötigte Zeit (inkl. der Wartezeit auf die Paging-Durchführung).

Modell 3: Fehlerentdeckung durch Konsistenzüberprüfung

- Bei diesem Verfahren wird nach Ablauf einer gewissen Bearbeitungszeit (bspw. einer CPU-Zeitscheibe) eine Konsistenzprüfung des **gesamten** Auftrags vorgenommen. Dies kann durch Software in Form von Akzeptanztests oder durch spezielle Diagnose-Routinen geschehen /RAND 75/, /CLSK 87/. Die Zeitpunkte einer solchen Überprüfung werden **checkpoints** genannt. Dabei kann unterschieden werden, ob ein Verfahren ausschließlich den (Haupt-)Speicher oder auch die korrekte Ausführung des gesamten Auftrags prüft. In diese zweite Kategorie fallen die meisten Konsistenzchecks, die per Hardware in fehlertoleranten Rechensystemen vorgenommen werden. Ein häufig benutztes Verfahren ist z.B. das Vergleichen bzw. **Votieren** mehrfach vorhandener, identischer Kopien eines Auftrags, die auf diversitärer Hardware abgearbeitet werden. In solchen votierenden Systemen muß durch das Betriebssystem gewährleistet sein, daß bei der Entdeckung eines nicht maskierbaren Fehlers der Auftrag ab dem letzten checkpoint wiederholt werden kann. Wenn selten votiert wird, kann die Latenzzeit groß werden, so daß relativ aufwendige Maßnahmen der Fehlerbehandlung, insbesondere zur Wiederherstellung der Datenintegrität, erforderlich sind.

Fehlerbehandlung:

- Weil die Fehlerentdeckung erst am Ende eines Bearbeitungsabschnittes geschieht, haben die betroffenen Aufträge u.U. vor der Fehlerentdeckung verfälschte Daten benutzt und müssen deshalb ab dem letzten checkpoint wiederholt werden. Es kann bei diesem Verfahren nicht festgestellt werden, welche der Seiten eine physikalische Störung erlitten haben, so daß alle in

diesem Bearbeitungsabschnitt benutzten Seiten neu eingelagert werden müssen. Dieses Verfahren ist der Fehlerentdeckung beim Seitenauslagern sehr ähnlich. Allerdings sind hier die Zeitpunkte einer Seitenüberprüfung vom Seitenwechselverhalten weitgehend unabhängig; sie finden stets nach festen Ausführungsdauern statt, die durch den Benutzer oder das Betriebssystem bestimmt sind.

Zu den drei oben genannten Verfahren der Fehlerentdeckung können noch einige gemeinsame Bemerkungen gemacht werden.

- Bei Modell 1 oder 3 wird ein Fehler erst relativ spät festgestellt, sodaß u.U. eine Seite auch mehrere Störungen erfahren haben kann. Bei diesen Verfahren wird dann nur erkannt, daß ein (Seiten-) Fehler aufgetreten ist. Es kann jedoch keine Aussage darüber getroffen werden, wieviele Störungen die betroffene Seite während der Latenzzeit erlitten hat.

- Bei den Modellen 2 und 3 muß berücksichtigt werden, wie sich Störungen auswirken, die nach dem letzten Zugriff, d.h. direkt vor dem Auslagern einer Seite aufgetreten sind. In manchen Speicherverwaltungssystemen führt eine Störung zu keinem Fehler, wenn die betroffene Seite vor einem erneuten Zugriff ausgelagert wird. Dies ist realistisch, wenn im Speicherverwaltungs-System jede Seiten-Änderung augenblicklich auf den Hintergrundspeicher "**durchgeschrieben**" wird, und so kein Rückschreiben der Seite bei ihrer Auslagerung erforderlich ist.

Um in den nächsten Abschnitten **stochastische Modelle** zur Entdeckung und Behandlung von Speicherfehlern entwickeln zu können, gehen wir für alle drei genannten Verfahren der Fehlerentdeckung von den folgenden Annahmen aus:

Modellvoraussetzungen

- Bei allen **Techniken der Fehlerentdeckung** (Prüfsummenbildung, fehlererkennende Codes, Konsistenzchecks) gibt es eine gewisse, sehr kleine **Restfehlerwahrscheinlichkeit**, mit der bestimmte Fehlerklassen nicht erfaßt werden können (z.B. 3-Bit-Fehler bei Hamming-Codes). Wir wollen diese Fragestellung aber in dieser Arbeit ausschließen und annehmen, daß alle aufgetretenen Verfälschungen durch die betreffenden Verfahren erkannt werden. Zwar ist eine solche 100% Fehlerüberdeckung unrealistisch, aber mit erhöhtem Aufwand, bspw. noch mehr Coderedun-

danz, kann eine Restfehlerwahrscheinlichkeit von nahezu Null erreicht werden /CHHS 84/.

- Jede Seite s_i habe in Phase I die **(stationäre) Zugriffswahrscheinlichkeit** $\pi_{i(I)}$*. Es sollen also für jede Phase I, mit I=1,..,L, die Voraussetzungen des **IRM-Modells (independent reference model)** gelten, siehe Abschnitt 2.1.

- Der **Zugriff** auf Speicherseiten erfolge in Phase I mit der **Rate** β_I. Diese Rate beschreibt das Speicherverhalten des Auftrags und ist von der Gesamtlast des Systems unabhängig.

Zur Notation

- In den nächsten Abschnitten dieses Kapitels werden wir stets eine **beliebige aber feste Phase** I betrachten. **Sämtliche** im folgenden betrachteten **Größen beziehen sich auf** diese **Phase I.** Im weiteren wollen wir, wenn die Gefahr einer Verwechslung nicht besteht, den Phasen-Index I weglassen, um die Notation zu vereinfachen. (D.h. statt $\pi_{i(I)}$ und β_I werden wir π_i und β verwenden).

- In den Abschnitten 5.2 bis 5.4. betrachten wir jeweils eines der drei oben genannten Verfahren zur Fehlerentdeckung. Die dabei bestimmten Größen beziehen sich dann immer auf das im jeweiligen Abschnitt betrachtete Verfahren. Bei der Notation wird dies dadurch kenntlich gemacht, daß die vom **Verfahren der Fehlerentdeckung** abhängigen Größen einen **entsprechenden Index** j, mit $j \in \{1,2,3\}$ erhalten. So ist $C_{i,1}$ die Anzahl der Seitenüberprüfungen für die Fehlerentdeckung beim Seitenauslagern, $C_{i,2}$ diejenige für eine Fehlerentdeckung beim Seitenzugriff usw. Wenn in einem Abschnitt von nur einem Verfahren die Rede ist und die Gefahr einer Verwechselung nicht besteht, wird im weiteren dieser Index **weggelassen.**

Nach diesen Vereinbarungen, wollen wir jetzt noch eine für alle Verfahren der Fehlerentdeckung grundlegende Größe ableiten.
Die Wahrscheinlichkeit, mit der eine Seite s_i von `ereignisunabhängigen` Störungen betroffen ist, hängt davon ab, wie lange diese Seite zwischen zwei Fehlerüberprüfungen im Hauptspeicher eingelagert ist, vgl. Abschnitt 4.1.1. Die genannten Verfahren der Fehlerentdeckung unterscheiden sich offensichtlich dadurch, daß in verschiedenen Zeitabständen geprüft wird, ob Störungen aufgetreten sind.

Sei s_i eine beliebige aber feste Seite aus Phase I, dann benötigen wir für alle Modelle die **Verweildauer** dieser Seite im Arbeitsspeicher $V_{i(I)}$,

abgekürzt durch V_i. Sie errechnet sich wie folgt aus der mittleren Gesamt-Verweildauer V_I des Auftrags in Phase I, die im folgenden mit V bezeichnet wird. (Mit der Berechnung der Verweildauer V beschäftigen wir uns im Kapitel 6.)

Berechnung der Verweildauer einer Seite

Bei einer Fenstergröße T ergibt sich für die Wahrscheinlichkeit a_i, mit der sich die Seite s_i in der Arbeitsmenge, d.h. im Hauptspeicher, befindet:

$$a_i = 1 - (1 - \pi_i)^T \quad . \qquad (5.1\text{-}1)$$

Der Subtrahend $(1 - \pi_i)^T$ bezeichnet die Wahrscheinlichkeit, daß bei den letzten T Zugriffen Seite s_i nicht referenziert wurde. Die Wahrscheinlichkeit a_i soll im weiteren **Aufenthaltswahrscheinlichkeit** (der Seite s_i) genannt werden.

- In Gleichung (5.1-1) wird davon ausgegangen , daß jede bei den letzten T Zugriffen nicht referenzierte Seite auch wirklich auf den Hintergrundspeicher ausgelagert wird. Diese Annahme ist dann realistisch, wenn das Dispatching-System gut ausgelastet ist, d.h. so viele Aufträge aktiv sind, daß für deren Arbeitsmengen der gesamte Hauptspeicher benötigt wird. Nur unter dieser **heavy-load-Annahme** gilt in der Praxis auch das working-set-Prinzip ohne Einschränkung: daß sich nämlich nur die Seiten der letzten T Zugriffe im Hauptspeicher befinden.

Mit Hilfe von a_i ergibt sich für die **mittlere Verweildauer der Seite s_i** im Hauptspeicher:

$$\begin{aligned} \overline{V}_i &= a_i \cdot \overline{V} \\ &= [1 - (1 - \pi_i)^T] \cdot \overline{V} \\ &= \overline{V} - (1 - \pi_i)^T \cdot \overline{V} \quad , \end{aligned} \qquad (5.1\text{-}2)$$

mit der mittleren Gesamtverweildauer $\overline{V}$ des Auftrags in Phase I. In Analogie zur Definition der Verfügbarkeit /SCHN 80/ kann also a_i als ein Zeitverhältnis aber auch als eine Wahrscheinlichkeit (wie in (5.1-1)) aufgefaßt werden.

Bemerkung
Die Verweildauer eines Auftrags kann natürlich als Zufallsvariable mit entsprechender Verteilungsfunktion aufgefaßt werden. Leider ist es mit den Standard-Lösungsverfahren für Warteschlangen-Modelle /LAVE 83/, /BEIL 88/, /BOAK 82/ (vgl. Kapitel 6 in dieser Arbeit) nicht möglich,

die Verteilungsfunktion von V in unserem Modell zu bestimmen, sodaß wir uns an dieser Stelle, wie im weiteren Verlauf dieses Kapitels mit **Mittelwert-Betrachtungen** begnügen müssen. Dies ist natürlich eine wesentliche Einschränkung unserer Modellierung, die uns aber andererseits eine einfache Ableitung der gewünschten Modellgrößen erlaubt.

In der folgenden Abbildung 5.1 wird die Abhängigkeit der relativen Verweildauer $\overline{V}_i$ (bezogen auf $\overline{V}$) von der Zugriffswahrscheinlichkeit π_i bei verschiedenen Fenstergrößen T veranschaulicht.

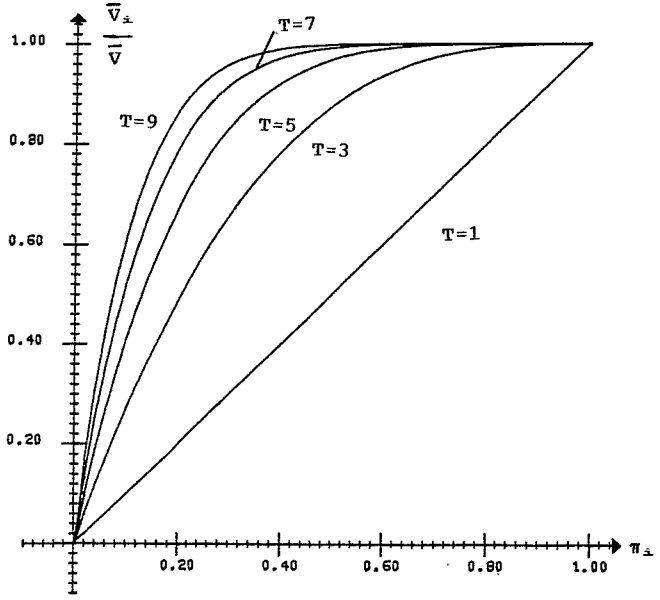


Abb. 5.1: Abhängigkeit der normierten Verweildauer $\overline{V}_i/\overline{V}$ von der Zugriffswahrscheinlichkeit π_i bei verschiedenen Fenstergrößen T.

Es fällt dabei insbesondere der exponentielle Anstieg der Verweildauer mit zunehmender Zugriffswahrscheinlichkeit π_i auf. Je größer die vom Betriebssystem benutzte Fenstergröße ist, umso schneller wächst mit ansteigender Zugriffswahrscheinlichkeit die Verweildauer einer Seite. Vor allem bei kleinen Zugriffswahrscheinlichkeiten sind für verschie-

dene Fenstergrößen signifikante Unterschiede zwischen den Verweildauern, und somit auch den Aufenthaltswahrscheinlichkeiten, festzustellen.

Nachdem die für alle Verfahren der Fehlerentdeckung erforderliche Verweildauer einer Seite bestimmt wurde, wollen wir nun im nächsten Abschnitt die Fehlerentdeckung beim Auslagern untersuchen.

5.2 Fehlerentdeckung bei der Seitenauslagerung

In diesem Abschnitt wird vorausgesetzt, daß bei jeder **Seitenauslagerung**, z.B. mit Hilfe eines fehlererkennenden Codes /CHHS 84/ oder mit Prüfsummenverfahren /DIGE 87/, überprüft wird, ob die Seite verfälschte Speicherworte enthält.
Nicht Gegenstand dieses Abschnittes sind hingegen die beim **Einlagern** von Seiten entdeckten Fehler. Diese sind nicht im Hauptspeicher, sondern im Hintergrundspeicher oder bei der Daten-Übertragung entstanden. Es handelt sich deshalb um die in Abschnitt 4.2 eingeführten Paging-Fehler, die in unserem Modell durch ereignisabhängige Störungen verursacht werden; Paging-Fehler werden in Abschnitt 5.5 betrachtet.

Bevor wir jetzt mit unserer Modellentwicklung beginnen, wollen wir noch einige weitere **Voraussetzungen** festlegen, die speziell in unser Modell für eine Fehlerentdeckung beim Auslagern eingehen.

- Jede betrachtete Seite s_i wird bei ihrem ersten Zugriff aus dem Hintergrundspeicher in den Hauptspeicher eingelagert (demand-paging-Prinzip).

- Eine Seite s_i wird aus dem Hauptspeicher ausgelagert, sobald sie bei den letzten T Seitenzugriffen nicht referenziert worden ist. D.h. wir gehen davon aus, daß das **working-set-Prinzip ohne Einschränkungen** gilt. In realen Systemen werden Seiten u.U. nicht so schnell ausgelagert, wenn nämlich der (eigentlich zuviel) belegte Speicherplatz nicht von anderen Aufträgen benötigt wird. Diese Voraussetzung impliziert demnach eine heavy-load-Annahme (siehe Abschnitt 5.1).

- Eine offensichtliche Folgerung der heavy-load-Annahme ist, daß mit der Terminierung eines Auftrags alle in seiner Arbeitsmenge befindlichen Seiten ausgelagert werden.

- Zur Vereinfachung unseres Modells nehmen wir an, daß alle Seiten, die aus der Arbeitsmenge ausgelagert werden, auch wirklich auf den Hintergrundspeicher zurückgeschrieben und dabei überprüft werden. In vielen Speicherverwaltungen wird durch ein sogenanntes **dirty bit** /DUNK 89/ zwischen beschriebenen und unveränderten Seiten unterschieden; unveränderte Seiten müssen nicht auf den Hintergrundspeicher zurückgeschrieben werden. In unserem Modell setzen wir voraus, daß wirklich jede Seite beim Auslagern überprüft wird, andernfalls könnten unentdeckte Verfälschungen auftreten.

Auch im folgenden müssen wir **ereignisunabhängige** von **ereignisabhängigen** Störungen (vgl. Kapitel 4) unterscheiden. Zunächst wollen wir ereignisunabhängige Störungen betrachten, deren Anzahl von den Verweildauern der Seiten im Hauptspeicher abhängen.

Ereignisunabhängige Fehler

Nach Abschnitt 4.1 werden ereignisunabhängige Störungen mit Hilfe eines Erneuerungsprozesses modelliert. Zur Untersuchung ereignisunabhängiger Fehler wollen wir von einer einzelnen Seite s_i in einer bestimmten Phase I ausgehen.

Zunächst werden wir die **Wahrscheinlichkeit** berechnen, **mit der ein Fehler beim Auslagern von s_i entdeckt wird.**
Dazu benötigen wir die Zeitdauer, die sich eine bestimmte Seite s_i seit der letzten Fehlerüberprüfung im Arbeitspeicher befunden hat; denn neue, noch nicht entdeckte die Seite s_i betreffende Störungen können nur in dieser Zeit aufgetreten sein. Gesucht ist also der Zeitraum zwischen der Einlagerung und der Auslagerung von s_i. Er soll im weiteren **Einlagerungsdauer D^E_i** genannt werden.

Berechnung der mittleren Einlagerungsdauer:

Zur Berechnung von D^E_i muß zunächst die Anzahl der Zugriffe auf Seite s_i bestimmt werden, denn mit dieser Größe läßt sich die Anzahl der Ein-/Auslagerungen von s_i abschätzen. Die mittlere Anzahl aller Zugriffe Z_i auf s_i läßt sich mit der **Zugriffsrate ß** berechnen. Auf Seite s_i wird nämlich mit der Rate

$$ß_i := ß_{i(I)} = \pi_i \cdot ß \qquad (5.2\text{-}1)$$

zugegriffen. Es finden also während der Bediendauer B_I in Phase I im Mittel

$$\overline{Z}_i = ß_i \cdot B_I = \pi_i \cdot (ß \cdot B_I) \qquad (5.2\text{-}2)$$

Zugriffe auf Seite s_i statt. Die **Bediendauer B_I** entspricht der CPU-Bedienzeitanforderung des Auftrags in Phase I. Mit Hilfe der Wahrscheinlichkeit π_I, mit der sich ein Auftrag in Phase I befindet (2.1-10), erhält man offensichtlich:

$$B_I = \pi_I \cdot B^G \ , \qquad (5.2\text{-}3)$$

wenn B^G die gesamte Bedienzeitanforderung des Auftrags ist.

Zur Vereinfachung der Notation werden wir im weiteren für die Bedienzeitanforderung des Auftrags in der Phase I nur B schreiben.

Die mittlere Einlagerungsdauer einer Seite s_i ist der Anteil ihrer Verweildauer V_i (siehe 5.1-2), der auf eine einzelne Einlagerung entfällt.
Die Anzahl der Einlagerungen einer Seite ist gleich der **Anzahl** ihrer Auslagerungen bzw. **der Fehlerüberprüfungen** C_i dieser Seite.

Wenn kein Fehler aufgetreten ist, wird die Seite s_i zum einen dann ausgelagert, wenn sie bei den letzten T Zugriffen nicht mehr referenziert wurde. (Hier geht die heavy load Annahme ein.) Die Wahrscheinlichkeit, daß auf die Seite T-mal hintereinander nicht zugegriffen wird, ist analog zu (5.1-1) durch $(1-\pi_i)^T$ gegeben. D.h. die mittlere Anzahl dieser Auslagerungen der Seite s_i errechnet sich, indem man die Wahrscheinlichkeit $(1-\pi_i)^T$ mit der Anzahl $\overline{Z}_i$ aller Zugriffe auf s_i multipliziert.
Eine Seite s_i wird zum andern auch dann ausgelagert und überprüft, wenn sie sich beim Bearbeitungsende des Auftrags noch im Speicher befindet. Die Wahrscheinlichkeit für diesen Fall ist aber genau die Aufenthaltswahrscheinlichkeit a_i gemäß Gleichung (5.1-1). Insgesamt ergibt sich damit nach dem Satz vom totalen Erwartungswert /SCHN 80/:

$$\begin{aligned}\overline{C}_i &= \overline{Z}_i \cdot (1-a_i) + 1 \cdot a_i \\ &= \pi_i \cdot (1-\pi_i)^T \cdot \beta \cdot B + [1 - (1-\pi_i)^T] \\ &= 1 + (1-\pi_i)^T \cdot (\pi_i \cdot \beta \cdot B - 1) \ . \end{aligned} \qquad (5.2\text{-}4)$$

Die letzte Zeile von Gleichung (5.2-4) läßt sich leicht folgendermaßen interpretieren: Der erste Summand ("1") steht für den ersten Zugriff auf Seite s_i, denn dieser führt stets zu ihrer Einlagerung. Der zweite Summand enthält die Anzahl aller restlichen Einlagerungen von s_i. Insgesamt haben nämlich weitere $(\pi_i \cdot \beta \cdot B - 1)$ Zugriffe auf s_i stattgefunden, wobei mit der Wahrscheinlichkeit $(1-\pi_i)^T$ in den letzten T Speicherzugriffen nicht auf s_i zugegriffen worden ist.

Durch (5.2-1) bis (5.2-4) ist die mittlere Anzahl der Ein- bzw. Auslagerungen von Seite s_i bestimmt. Dies ist gleichzeitig auch die mittlere Anzahl der Fehlerüberprüfungen, die an Seite s_i vorgenommen werden. Sie wird in Abbildung 5.2 in Abhängigkeit von der Zugriffswahrscheinlichkeit und der Fenstergröße dargestellt.

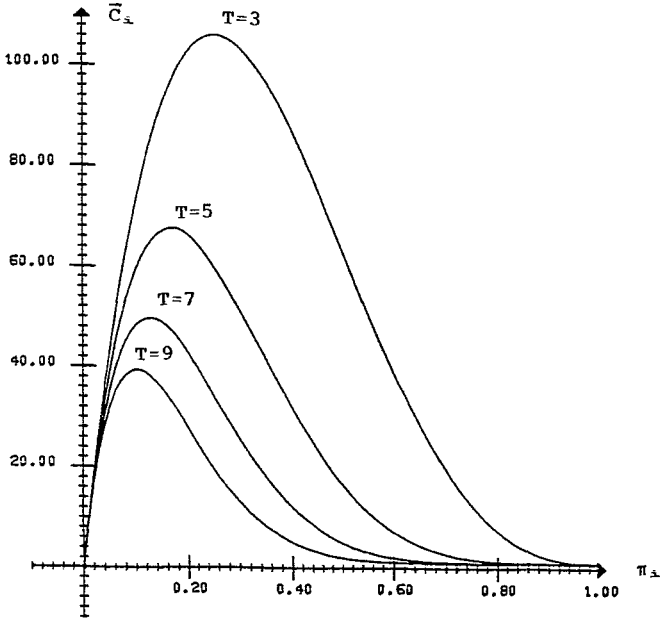


Abb. 5.2: Mittlere Anzahl der Seitenauslagerungen in Abhängigkeit von der Zugriffswahrscheinlichkeit bei verschiedenen Fenstergrößen T (mit ß=100, B=10).

Man erkennt, daß erwartungsgemäß die Anzahl der Auslagerungen mit steigender Fenstergröße geringer wird. Dabei wird je nach Fenstergröße bei Zugriffswahrscheinlichkeiten zwischen 0.1 und 0.3 ein Maximum erreicht. Seiten mit sehr großer Zugriffswahrscheinlichkeit befinden sich fast ständig im Arbeitsspeicher (vgl. Abb. 5.1) und werden nur einmal bei Terminierung des Auftrags ausgelagert
($\lim C_i = 1$, für $\pi_i \to 1$).

Finden $\bar{C}_i$ Ein-/Auslagerungen der Seite s_i statt, dann kann die mittlere Zeitdauer $\bar{D}^E_i$, die auf eine dieser $\bar{C}_i$ Einlagerungen entfällt, durch:

$$\bar{D}^{E_i} \approx \bar{V}_i / \bar{C}_i$$

$$= \frac{[1 - (1-\pi_i)^T] \cdot \bar{V}}{\pi_i \cdot (1-\pi_i)^T \cdot B \cdot \beta + [1 - (1-\pi_i)^T]} \qquad (5.2\text{-}5)$$

abgeschätzt werden. In Gleichung (5.2-5) bleiben etwaige zusätzliche Auslagerungen der Seite s_i (aufgrund von Fehlern) unberücksichtigt. Sie werden am Ende dieses Abschnitts (5.2-29) in Betracht gezogen.

Die mittlere Einlagerungsdauer $\bar{D}^{E_i}$ wird in der folgenden Abbildung bezogen auf $\bar{V}$ dargestellt. Erwartungsgemäß steigt die Einlagerungsdauer mit der Zugriffswahrscheinlichkeit und der Fenstergröße an.

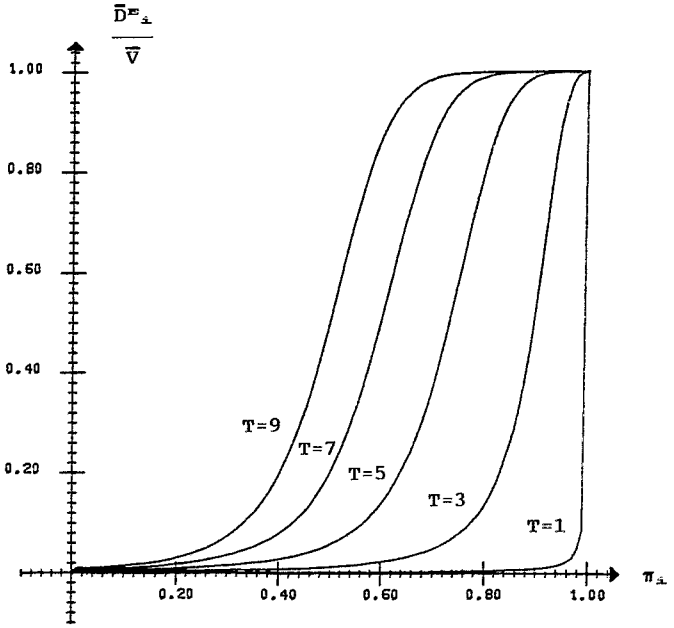


Abb. 5.3: **Die normierte mittlere Einlagerungsdauer einer Seite $\bar{D}^{E_i}/\bar{V}$ in Abhängigkeit von der Zugriffswahrscheinlichkeit π_i bei verschiedenen Fenstergrößen T.**

Berechnung der Wahrscheinlichkeit, mit der die Seite s_i gestört wird

Zwischen zwei Fehlerüberprüfungen befindet sich s_i im Mittel für den Zeitraum $\overline{D}^E_i$ im Speicher. Damit kann nun die **Wahrscheinlichkeit** berechnet werden, **mit der** eine bestimmte Seite s_i im Arbeitsspeicher vor ihrer Auslagerung **gestört worden ist**, bzw., mit der bei einer Fehlerüberprüfung ein Fehler entdeckt wird. Zunächst bestimmen wir dazu die Anzahl aller ereignisunabhängigen Störungen, die während der Einlagerungsdauer $\overline{D}^E_i$ auftreten.

Sei ein ereignisunabhängiger Störungsprozeß nach Abschnitt 4.1 vorausgesetzt, der durch einen Erneuerungsprozeß modelliert wird. Wird eine Fehlerüberprüfung beim Auslagern vorgenommen, dann müssen diejenigen Störungen betrachtet werden, die während der Einlagerungsdauer aufgetreten sind. Es ergibt sich das folgende Bild:

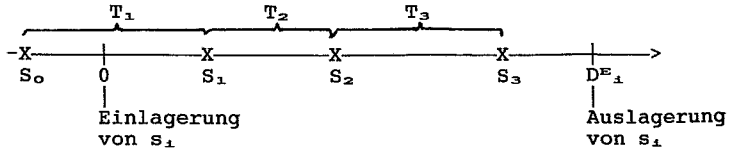


Abb 5.4: Ereignisunabhängiger Störungsprozeß während der Einlagerungsdauer D^E_i einer Seite s_i.

In Abbildung 5.4 sind die Störungszeitpunkte S_i so durchnummeriert, daß S_1 den ersten Störungszeitpunkt seit der Einlagerung von Seite s_i bezeichnet.

Die Verteilung der **Anzahl** N^U_i aller während der mittleren Einlagerungsdauer $\overline{D}^E_i$ von Seite s_i im gesamten Hauptspeicher aufgetretenen **ereignisunabhängigen Störungen** kann, wie in Abschnitt 4.1 dargestellt, mit Hilfe der Erneuerungstheorie berechnet werden. Nach Gleichung (4.1-9) gilt:

$$\text{Prob}[N^U_i=k] = Pr^U(k,\overline{D}^E_i)$$

$$= F^V_{Si}(\overline{D}^E_i) - F^V_{Si+1}(\overline{D}^E_i). \qquad (5.2\text{-}6)$$

Bei den oben gemachten Überlegungen ist zu beachten, daß wir die mittlere Einlagerungsdauer $\overline{D}^E_i$ zugrunde gelegt haben, weil es uns aus den in Abschnitt 5.1 angeführten Gründen nicht möglich ist, die Verteilungsfunktion von D^E_i zu bestimmen. Wäre die Verteilungsdichte $f^E_i(t)$ der Einlagerungsdauer gegeben, so erhielte man:

$$\text{Prob}[N^{U}_{i}=k] = \int_{0}^{\infty} Pr^{U}(k,t)\cdot f^{E}_{i}(t) \quad . \tag{5.2-7}$$

Da die $Pr^{U}(k,\overline{D}^{E}_{i})$ aus Gleichung (5.2-6) u.U. sehr schlecht zu berechnen sind, ist es oft hilfreich, den Mittelwert der Störungsanzahl mit dem mittleren Erneuerungsabstand $\overline{s}$, d.h. der mittleren Zeitdauer zwischen zwei Erneuerungszeitpunkten, zu bestimmen. Der mittlere Erneuerungsabstand $\overline{s}$ ist durch (4.1-11) bzw. bei exponential verteilten Störungsabständen durch (4.1-13) gegeben.

Mit Hilfe von $\overline{s}$ ergibt sich für $\overline{N}^{U}_{i}$ die mittlere Anzahl der ereignisunabhängigen Störungen, die während der Einlagerungsdauer $\overline{D}^{E}_{i}$ von Seite s_i aufgetreten sind:

$$\overline{N}^{U}_{i} = \overline{D}^{E}_{i}/\overline{s}$$

$$= \frac{[1 - (1-\pi_i)^{T}]\cdot\overline{V}}{\pi_i\cdot(1-\pi_i)^{T}\cdot B\cdot\beta - (1-\pi_i)^{T} + 1}\cdot\overline{s}^{-1} \quad . \tag{5.2-8}$$

Unser Ziel war jedoch nicht nur die Anzahl der Störungen, die im gesamten Hauptspeicher aufgetreten sind, zu bestimmen, sondern vornehmlich, mit welcher Wahrscheinlichkeit eine bestimmte Seite s_i von einer dieser Störungen betroffen ist.
Dazu muß die Auswirkung einer Störung, d.h. die Anzahl der von ihr betroffenen Seiten berücksichtigt werden. Diese wurde in Abschnitt 4.1.2 untersucht und ist durch die Verteilung der e(k), $1\leq k\leq H$, gemäß (4.1-19) gegeben.

Von jeder Störung sind nach (4.1-20) im Mittel $\overline{e}$ Seiten betroffen, d.h. eine der insgesamt H Hauptspeicherseiten wird mit der Wahrscheinlichkeit

$$p_s = \overline{e}/H \tag{5.2-9}$$

von einer einzelnen Störung verfälscht. Während der Einlagerungsdauer ist aber s_i nicht nur einem Störereignis ausgesetzt, sondern nach (5.2-6) bzw. (5.2-8) finden N^{U}_{i} Störereignisse im gesamten Hauptspeicher statt. Die Wahrscheinlichkeit q^{U}_{i}, daß Seite s_i von mindestens einer der N^{U}_{i} aufgetretenen Störungen betroffen wurde, ist dann die Wahrscheinlichkeit der Vereinigungsmenge dieser N^{U}_{i} Ereignisse. Sie kann nach der Formel von Poincaré-Sylvester /SCHN 80/ berechnet werden, die sich hier vereinfacht, weil der Spezialfall gleichwahrscheinlicher Elementarereignisse gegeben ist. Berücksichtigt man gemäß (5.2-6) die Verteilung $Pr^{U}(k,\overline{D}^{E}_{i})$ der Störungs-Anzahl, so erhält man für die **Wahrscheinlichkeit einer Fehlerentdeckung:**

$$q^{U}_{i} = \sum_{k=1}^{\infty} [Pr^{U}(k,\bar{D}^{E}_{i}) \cdot \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{j} \cdot p_{s}^{j}] \quad . \qquad (5.2\text{-}10)$$

Bemerkung:
In unserem Modell betrachten wir ausschließlich Fehler innerhalb einer ganzen Seite. Dies ist realistisch, weil bei vielen Verfahren der Fehlererkennung beim Auslagern (z.B. Block-Codes /CHHS 84/) nur eine Aussage darüber gemacht werden kann, ob eine Störung innerhalb der Seite stattgefunden hat und nicht darüber, welches Speicherwort verfälscht worden ist. Ebenso werden Speicherfehler normalerweise dadurch behoben, daß ganze Seiten neu eingelagert werden. D.h. es ist meist unerheblich, wieviele Wörter der Seite verfälscht worden sind; die erforderlichen Aktionen zur Fehlerbehebung sind von der Anzahl der innerhalb einer Seite verfälschten Worte unabhängig.

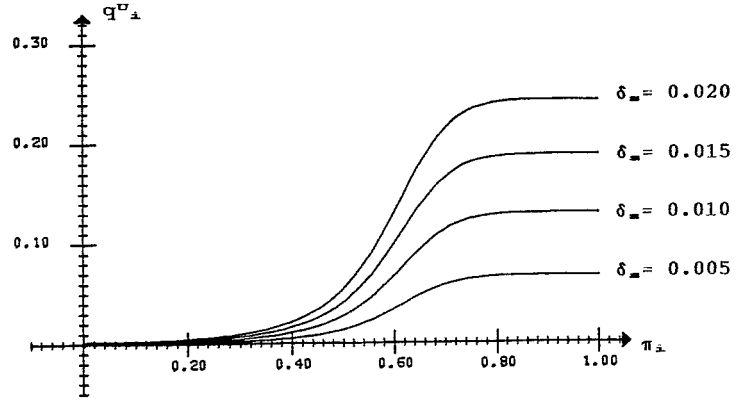


Abb. 5.5: Die Wahrscheinlichkeit q^{U}_{i} für die Entdeckung einer ereignisunabhängigen Störung bei verschiedenen Poissonschen Auftritts-Raten (und gewählter fester Fenstergröße T=7, V=1000, p_S=0.014).

In Abbildung 5.5 wird q^{U}_{i} dargestellt. Dabei wurde angenommen, daß N^{U}_{i} gemäß (4.1-14) mit der Rate δ_s poisson-verteilt ist. Man erkennt, daß sich in unserem Beispiel erst ab einer Zugriffswahrscheinlichkeit von etwa π_i=0.4 signifikante Unterschiede für verschiedene Störungs-Raten ergeben. Ab einer Zugriffswahrscheinlichkeit von π_i=0.8 ist für ein festes δ_S die Wahrscheinlichkeit q^{U}_{i} einer Fehlerentdeckung beim Auslagern konstant. Dies ist bei einer gewählten Fenstergröße T=7 mit Abbildung 5.3 konsistent, denn ab diesem Wert ist die Einlagerungdauer ungefähr gleich der Verweildauer.

Bei (5.2-10) muß natürlich beachtet werden, daß in diese Wahrscheinlichkeit auch die Verweildauer $\overline{D}^{F_i}$ eines Auftrags durch die Größe $Pr^U(k,\overline{D}^{E_i})$ eingeht. Wegen der zur Fehlerbehandlung erforderlichen zusätzlichen Seiteneinlagerungen wächst aber wiederum die Verweildauer mit der Fehlerwahrscheinlichkeit q^{U_i}. Diese rekursive Abhängigkeit konnte in den genannten Gleichungen nicht berücksichtigt werden. Sie wird aber durch ein iteratives Verfahren (Kapitel 7) nachgebildet.
Abschließend interessiert uns natürlich auch, die mittlere **Anzahl** $\overline{N}^{FU_i}$ der bei einer Seite s_i durch **ereignisunabhängige** Störungen verursachten **Fehler**. Dazu muß offensichtlich die Anzahl ihrer Fehlerüberprüfungen $\overline{C}_i$ mit der Wahrscheinlichkeit q^{U_i} einer Fehlerentdeckung multipliziert werden.

$$\overline{N}^{FU_i} = \overline{C}_i \cdot q^{U_i} \quad . \tag{5.2-11}$$

$\overline{N}^{FU_i}$ ist in Abbildung 5.6 dargestellt, wobei dieselben Voraussetzungen wie in Abbildung 5.5 gelten.

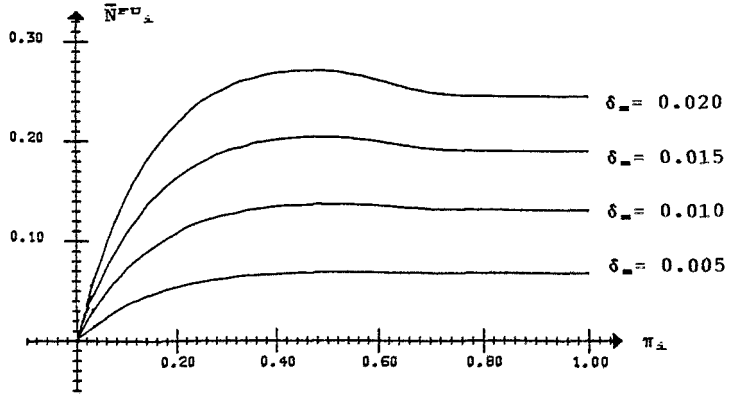


Abb. 5.6: Anzahl der entdeckten ereignisunabhängigen Fehler einer Seite in Abhängigkeit von ihrer Zugriffswahrscheinlichkeit bei verschiedenen Störungs-Raten.

Man erkennt, daß die mittlere Anzahl der während einer Auftragsausführung entdeckten Fehler mit zunehmender Zugriffswahrscheinlichkeit exponential ansteigt. Ab einer bestimmten Zugriffswahrscheinlichkeit (hier etwa π_i=0.50) sind die Kurven konstant. Dies liegt daran, daß

ab dieser Zugriffswahrscheinlichkeit auch $\overline{D}^{E_i}$ und $\overline{C}_i$ konstant sind, vgl. die Abbildungen 5.2, 5.3 und 5.5.
Bei einer hohen Fehlerrate (etwa $\delta_S > 0.001$) haben die Kurven ungefähr bei $\pi_i = 0.4$ ein Maximum. Bei diesen Werten von δ_S treten mit hoher Wahrscheinlichkeit mehrere Störungen während der Gesamt-Verweildauer des Auftrags statt, so daß ein häufiger stattfindendes Überprüfen (d.h. kleinere Zugriffswahrscheinlichkeit) zu mehr entdeckten Fehlern führt.

Zugriffsinduzierte Fehler

Während der Einlagerungsdauer können nach Abschnitt 4.2 auch zugriffsinduzierte Störungen auftreten.
Die mittlere **Gesamtzahl $\overline{Z}^{E_i}$, aller während $\overline{D}^{E_i}$ aufgetretenen Zugriffe** auf Seiten des virtuellen Adreßraums, errechnet sich durch:

$$\overline{Z}^{E_i} = \beta \cdot B \cdot \frac{\overline{D}^{E_i}}{\overline{V}} \quad . \tag{5.2-12a}$$

Mit der Gleichung (5.2-5) ergibt sich (unter Vernachlässigung der durch Fehler verursachten zusätzlichen Einlagerungen von si):

$$\overline{Z}^{E_i} \approx \frac{\beta \cdot B}{\overline{C}_i} \quad . \tag{5.2-12b}$$

Auf die betrachtete Seite s_i erfolgen dann während ihrer Einlagerungsdauer (also zwischen zwei Seiten-Überprüfungen) im Mittel

$$\overline{Z}^{C_i} = \pi_i \cdot Z^{E_i} \quad . \tag{5.2-13}$$

Zugriffe. Die Anzahl der Zugriffe auf Seite s_i hängt offensichtlich stark von ihrer Einlagerungsdauer $\overline{D}^{E_i}$ ab (vgl. auch Abb. 5.3) und ist in der nächsten Abbildung 5.7 dargestellt.

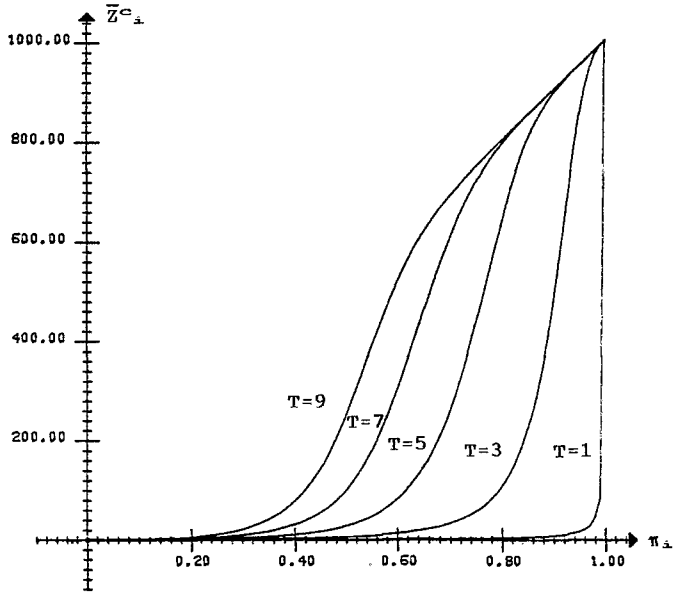


Abb.5.7: Anzahl der Zugriffe auf eine Seite s_i während ihrer Einlagerungsdauer in Abhängigkeit von ihrer Zugriffswahrscheinlichkeit π_i bei verschiedenen Fenstergrößen T.

Nach unserem Modell aus Abschnitt 4.2 induziert jeder Zugriff auf s_i mit der Wahrscheinlichkeit p_z einen Fehler, und die **Anzahl N^{z}_{i} zugriffsinduzierter Störungen** der Seite s_i ist dann binomisch verteilt. Man erhält also mit Gleichung (4.2-1):

$$\text{Prob}[N^{z}_{i}=k] = Pr^{z}(k,\overline{Z}^{c}_{i})$$

$$= \binom{\overline{Z}^{c}_{i}}{k} \cdot p_z^{\,k} \,(1 - p_z)^{\overline{Z}^{c}_{i}-k} \quad . \qquad (5.2\text{-}14)$$

Die mittlere Anzahl dieser Störungen ist dann:

$$\overline{N}^{z}_{i} = p_z \cdot \overline{Z}^{c}_{i} \quad . \qquad (5.2\text{-}15)$$

Uns interessiert aber nicht nur die Anzahl der aufgetretenen Störungen, sondern vielmehr die Wahrscheinlichkeit q^{z}_{i}, mit der während $\overline{D}^{E}_{i}$ mindestens eine Störung stattgefunden hat. D.h. die **Wahrscheinlichkeit,** mit der beim Auslagern eine **zugriffsinduzierte Störung** festgestellt wird. Sie ist gegeben durch:

$$q^{z}_{i} = 1 - (1 - p_z)^{\bar{Z}^{c}_{i}} \quad . \qquad (5.2\text{-}16)$$

In Gleichung (5.2-16) geht die realistische Annahme ein, daß sich jede zugriffsinduzierte Störung nur auf die Seite auswirkt, auf die zugegriffen wird, d.h., daß e(1)=1 gilt.
In den beiden folgenden Abbildungen ist q^{z}_{i} in Abhängigkeit von der Fenstergröße T (Abb. 5.8) sowie der Wahrscheinlichkeit p_z eines gestörten Zugriffs (Abb. 5.9) dargestellt.

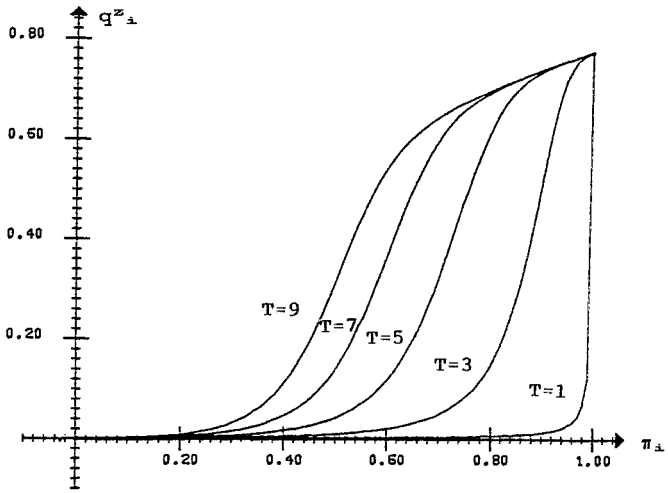


Abb. 5.8: Wahrscheinlichkeit, daß beim Auslagern eine zugriffsinduzierte Störung entdeckt wird in Abhängigkeit von verschiedenen Fenstergrößen T, wobei p_z=0.0015 gilt.

Ab einer bestimmten Zugriffswahrscheinlichkeit π_i ist die Einlagerungsdauer ungefähr gleich der Verweildauer. Danach zeigt die Wahrscheinlichkeit einer Fehlerentdeckung q^{z}_{i} (hier: bei der Fenstergröße T=7 etwa ab π_i=0.65) lineares Verhalten. Abbildung 5.9 zeigt die Abhängigkeit dieser Größe von der Wahrscheinlichkeit p_z, mit der ein einzelner Zugriff verfälscht wird.

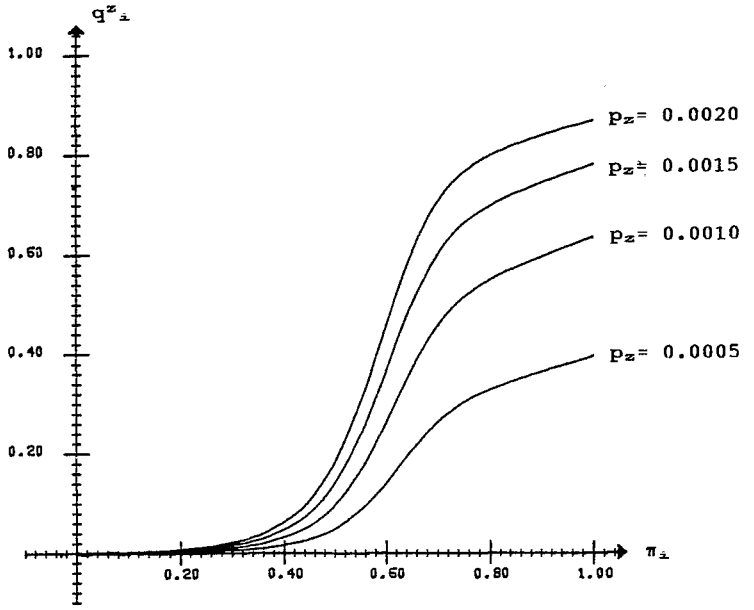


Abb. 5.9: **Wahrscheinlichkeit, daß beim Auslagern ein zugriffsinduzierter Fehler entdeckt wird in Abängigkeit von π_i und der Störungswahrscheinlichkeit eines Zugriffs p_z.**

Berechnung der mittleren Fehleranzahl in Seite s_i

Die oben berechneten Wahrscheinlichkeiten q^U_i und q^Z_i geben an, wie groß das Risiko ist, daß eine Seite s_i vor ihrer Auslagerung im Hauptspeicher gestört wird. Mit Hilfe dieser Größen läßt sich nun leicht die mittlere Anzahl der Fehler $\overline{N}^F_i$, die eine Seite s_i während ihrer Aufenthaltsdauer im Dispatching-System erleidet, errechnen.

Die Anzahl der durchgeführten Fehlerüberprüfungen von Seite s_i ist offensichtlich gleich der Anzahl ihrer Auslagerungen $\overline{C}_i$ und durch (5.2-4) gegeben. Bei jeder Überprüfung wird mit der Wahrscheinlichkeit q^U_i eine **ereignisunabhängige** und mit q^Z_i eine **zugriffsinduzierte** Störung entdeckt. Nach der Summenregel der Wahrscheinlichkeitsrechnung wird dann mit der Wahrscheinlichkeit q_i eine der beiden Störungs-Arten entdeckt:

$$q_i = q^U_i + q^Z_i - q^U_i \cdot q^Z_i \ . \qquad (5.2\text{-}17)$$

Die **Gesamt-Wahrscheinlichkeit einer Fehlerentdeckung** q_t wird in Abbildung 5.10 veranschaulicht. Dabei wurde einmal δ_m als Einflußgröße der zugriffsunabhängigen Störungen, das andere mal p_z als Parameter zugriffsinduzierter Störungen betrachtet.

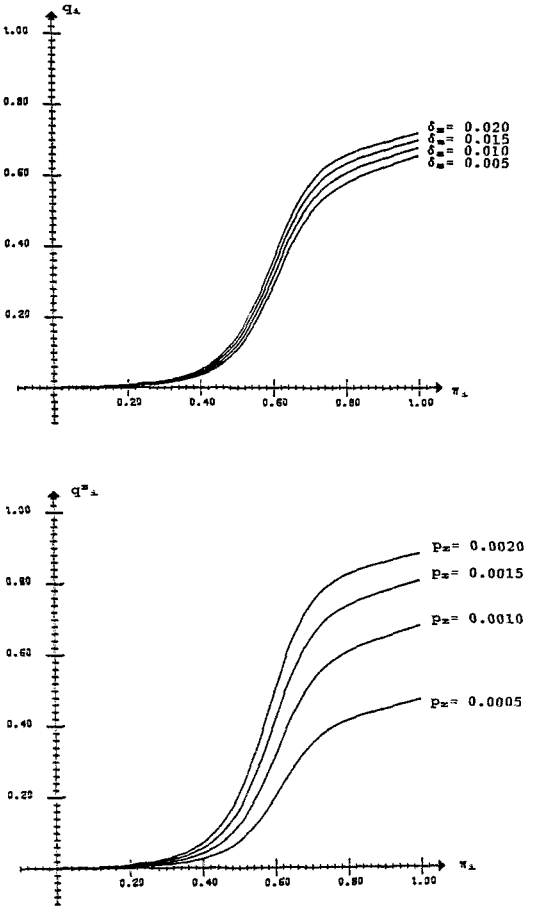


Abb.5.10: Gesamtwahrscheinlichkeit einer Fehlerentdeckung
a) bei verschiedenen Störungs-Raten δ_m
b) bei unterschiedlicher Störungs-Wahrscheinlichkeit p_z

Man erhält für die mittlere Anzahl der beim Auslagern von Seite s_i entdeckten Fehler:

$$\overline{N}^{F_i} = \overline{C}_i \cdot q_i$$

$$= \overline{C}_i \cdot (q^{U_i} + q^{Z_i} - q^{U_i} \cdot q^{Z_i}) \quad . \tag{5.2-18}$$

Die entsprechende Rate Φ^{F_i} "komprimiert" die Anzahl der aufgetretenen Fehler $\overline{N}^{F_i}$ auf die Bediendauer B des Auftrags, denn nur während der Ausführungsdauer des Auftrags können Fehler entdeckt werden:

$$\Phi^{F_i} = \overline{N}^{F_i}/B \quad . \tag{5.2-19}$$

Die Gesamt-Fehlerrate Φ^F ergibt sich dann durch Summation der Fehlerraten aller m virtuellen Seiten von Phase I:

$$\Phi^F = \sum_{1 \le i \le m} \Phi^{F_i} = [\sum_{1 \le i \le m} \overline{N}^{F_i}]/B \quad . \tag{5.2-20}$$

Durch Speicherfehler induzierte Last

Wir wollen nun die **Wirkungen** betrachten, die ein Speicherfehler hervorruft, d.h. die durch seine Behandlung induzierte Last.
Zur Behandlung eines beim Auslagern entdeckten Fehlers sind die folgenden Maßnahmen erforderlich:

- Der Auftrag muß ab dem Zeitpunkt wiederholt werden (**rollback**), zu dem die betroffene Seite eingelagert wurde. Als checkpoint /RAND 75/ kann der Zeitpunkt der letzten Einlagerung betrachtet werden, weil dann die Korrektheit der Seite sichergestellt war. Ein solches Verfahren setzt voraus, daß die für ein checkpointing erforderlichen Maßnahmen, z.B. Sicherung des Prozeß-Kontextes, durch das Betriebssystem vorgenommen werden. Andernfalls muß der gesamte Auftrag wiederholt werden.

- Die **betroffene Seite** muß erneut **eingelagert** werden.

- Um die **Datenintegrität** zu gewährleisten, müssen sämtliche Seiten, auf die während der Einlagerungsdauer der betroffenen Seite zugegriffen worden ist, erneut eingelagert werden. (Eigentlich müssten nur die in diesem Zeitraum beschriebenen Seiten betrachtet werden, denn nur sie können durch den aufgetretenen Fehler verfälscht worden sein; diese Unterscheidung wird aber in unserem Modell nicht gemacht.)

Wie mit Hilfe eines Cache-Speichers ein rollback implementiert werden kann, ist in /HUMA 87/ untersucht. Ein beim Auslagern von Seite s_i entdeckter Fehler verursacht also eine **zusätzliche Bediendauer**, und ist gleichzeitig mit **zusätzlichen Seiteneinlagerungen** verbunden.

Diese durch das Auftreten von Fehlern induzierte Systemlast muß in unser **Leistungsmodell** (siehe Kapitel 6) eingebunden werden. Dabei sind offensichtlich zwei verschiedene Teilprobleme zu betrachten:

- Wie verlängert sich in der betrachteten Phase durch Speicherfehler die Bediendauer B eines Auftrags ?

- In welchem Maße erhöht sich die Paging-Rate durch das erforderliche Einlagern verfälschter oder inkonsistenter Speicherseiten ?

Die durch Fehler induzierte Systemlast verursacht wiederum Rückwirkungen auf unser **Fehlermodell** (vgl. Abschnitt 2.2):

- Sie vergrößert die ins Fehlermodell eingehende Verweildauer des Auftrags.
- Die Anzahl der Einlagerungen einer Seite erhöht sich, um die Datenintegrität herzustellen. Diese durch Fehler verursachten zusätzlichen Einlagerungen einer Seite verkürzen auch die mittlere Einlagerungsdauer D^{E}_{i}.

Insgesamt erhöht die durch Fehler induzierte Last die Wahrscheinlichkeit eines Seitenfehlers noch weiter.

Verlängerung der Bediendauer B

Wird beim Auslagern ein Fehler in Seite s_i entdeckt, dann ist die im Dispatching-System zugebrachte "unnütze" Verweilzeit, d.h. die in der kein Auftragsfortschritt erzielt wurde, gleich der Einlagerungsdauer D^{E}_{i} von s_i. Ihr entspricht im Mittel eine Bearbeitungsdauer $\bar{B}^{R}_{i}$ von:

$$\bar{B}^{R}_{i} = \frac{\bar{D}^{E}_{i} \cdot B}{\bar{V}} \quad . \tag{5.2-21a}$$

Vernachlässigt man die durch Fehler verursachten Einlagerungen der Seite s_i (ihre Anzahl ist normalerweise um Größenordnungen kleiner als C_i) so läßt sich mit Gleichung (5.2-5) schreiben:

$$\bar{B}^{R}_{i} \approx B/\bar{C}_{i} \tag{5.2-21b}$$

Durch $\bar{B}^{R}_{i}$ ist also die mittlere Ausführungszeit gegeben, die beim Entdecken eines Fehlers in Seite s_i wiederholt werden muß, d.h. die mittlere Bearbeitungszeit, um die sich der Auftrag dann verlängert.
Betrachtet man alle Fehler, die sämtliche m virtuellen Seiten des Auftrags erleiden, dann ergibt sich mit $\bar{N}^{F}_{i}$ aus (5.2-18):

$$\bar{B}^{R} = \sum_{1 \le i \le m} q_i \cdot \bar{C}_i \cdot \bar{B}^{R}_{i}$$

$$= \sum_{1 \le i \le m} \bar{N}^{F}_{i} \cdot \bar{B}^{R}_{i} \quad , \tag{5.2-22a}$$

die durch Auftrags(-Teil)wiederholungen verursachte **zusätzliche mittlere Bediendauer eines Auftrags**. Wenn (5.2-21b) gilt, vereinfacht sich (5.2-22a) zu

$$\bar{B}^{R} \approx \sum_{1 \le i \le m} q_i \cdot a_i \cdot B \quad . \tag{5.2-22b}$$

Erhöhung der Paging-Rate

Die mittlere Anzahl aller bei Seite s_i erkannten Fehler ist durch Gleichung (5.2-18) gegeben. Durch Summation über alle m virtuellen Seiten erhält man die Gesamtzahl $\overline{N}^F$ aller beim Auslagern entdeckten Fehler:

$$\overline{N}^F = \sum_{1 \le i \le m} \overline{N}^F{}_i$$

$$= \sum_{1 \le i \le m} q_i \cdot \overline{C}_i \quad . \qquad (5.2\text{-}23)$$

In (5.2-23) bezeichnet $\overline{C}_i$ die Anzahl der "regulären" Auslagerungen, die Seite s_i erfährt, weil sie sich nicht mehr in der Arbeitsmenge befindet; $\overline{N}^F{}_i$ ist die Anzahl der bei s_i entdeckten Fehler.
Eine Seite s_i kann aber auch dann ausgelagert werden, wenn während ihrer Einlagerungsdauer bei einer anderen Seite der Arbeitsmenge ein Fehler entdeckt wird. Durch einen solchen Fehler kann die betrachtete Seite s_i während der Auftragsausführung mit fehlerhaften Daten beschrieben worden sein (error propagation), sodaß sie zur Wiederherstellung der Datenintegrität erneut eingelagert werden muß.

Im folgenden wird berechnet, wieviele Seiteneinlagerungen vorgenommen werden müssen, um die Datenintegrität zu gewährleisten. Dazu sei angenommen, daß ein Fehler bei der Auslagerung der Seite s_i entdeckt worden ist.
Insgesamt erfolgten während der Einlagerungsdauer von s_i, d.h. zwischen zwei Fehlerüberprüfungen, im Mittel

$$\overline{Z}^{C_i} = \beta \cdot \overline{B}^{R_i} \qquad (5.2\text{-}24)$$

Seitenzugriffe. Die Zugriffe referenzierten zum Teil dieselben Seiten. Um Datenintegrität zu erreichen, müssen alle Seiten erneut eingelagert werden, auf die innerhalb des genannten Zeitraums ein Zugriff stattfand. Gesucht ist also die mittlere Anzahl $\overline{\Gamma}^{C_i}$ der Seiten, die während der letzten $\overline{Z}^{C_i}$ Zugriffe referenziert wurden.

Analog zur Aufenthaltswahrscheinlichkeit a_i in der Arbeitsmenge (5.1-1) ist die Wahrscheinlichkeit, daß auf eine Seite s_j, $j \neq i$, während der letzten Z^{C_i} Zugriffe zugegriffen wurde, durch

$$1 - (1 - \pi_j)^{Z^{C_i}}$$

gegeben. Insgesamt ergibt sich dann für $\overline{\Gamma}^{C_i}$, die mittlere Anzahl der Seiten, die sich während der letzten Z^{C_i} Zugriffe vor der Auslagerung von s_i in der Arbeitsmenge befanden:

$$\bar{\Gamma}^{C_i} = 1 + \sum_{\substack{1 \le j \le m \\ j \neq i}} \{1 - (1 - \pi_j)^{\bar{Z}^{C_i}}\}$$

$$= m - \sum_{\substack{1 \le j \le m \\ j \neq i}} (1 - \pi_j)^{\bar{Z}^{C_i}} \quad . \qquad (5.2\text{-}25)$$

Die Frage nach der Anzahl der verschiedenen Seiten, die während der letzten $\bar{Z}^{C_i}$ Zugriffe benutzt wurden, entspricht dem Problem, die Arbeitsmengengröße bei einem virtuellen Speicherraum der Größe (m-1) mit der gegebenen Fenstergröße $\bar{Z}^{C_i}$ zu bestimmen. Deshalb ergibt sich mit (5.2-25) eine Formel, die zu der aus der Literatur bekannten konsistent ist /DESC 72/, vgl. auch Kapitel 6.

Die mittlere Anzahl aller durch Fehler verursachten zusätzlichen Seitenwechsel $\bar{\Gamma}^R$, erhält man, indem über die Fehlerentdeckungen aller Seiten summiert:

$$\bar{\Gamma}^R = \sum_{1 \le i \le m} (\bar{N}^F{}_i \cdot \bar{\Gamma}^{C_i})$$

$$= \sum_{1 \le i \le m} \bar{N}^F{}_i \cdot [m - \sum_{\substack{1 \le j \le m \\ j \neq i}} (1 - \pi_j)^{\bar{Z}^{C_i}}] \qquad (5.2\text{-}26)$$

Die entsprechende durch Hauptspeicherfehler induzierte zusätzliche Pagingrate Φ^{HF} beträgt dann:

$$\Phi^{HF} = \bar{\Gamma}^R / B \quad . \qquad (5.2\text{-}27)$$

Dies ist die Rate, um die sich die normale Paging-Rate des Speicherverwaltungssystems durch Speicherfehler erhöht.

Auswirkung von Fehlern auf die Einlagerungsdauer

In Gleichung (5.2-5) wurde die mittlere Einlagerungsdauer durch $\bar{D}^E{}_i = \bar{V}_i / \bar{C}_i$ bestimmt, wobei $\bar{C}_i$ die Anzahl der "regulären", d.h. nach dem demand-paging-Prinzip erfolgten, Einlagerungen der Seite s_i bezeichnet. Nach (5.2-23) treten aber insgesamt bei allen von s_i verschiedenen Seiten $\bar{N}^F - \bar{N}^F{}_i$ Fehler auf. Bei jedem dieser Fehler befindet sich s_i mit der Wahrscheinlichkeit a_i in der Arbeitsmenge. Die mittlere Anzahl $\bar{A}_i$ der Auslagerungen von s_i aufgrund von Fehlern anderer Seiten ist also:

$$\bar{A}_i = a_i \cdot \sum_{\substack{1 \le j \le m \\ i \neq j}} q_j \cdot \bar{C}_j \quad . \qquad (5.2\text{-}28)$$

Unter Einbeziehung dieser zusätzlichen Einlagerungen muß $\overline{D}^{E_i}$ wie folgt errechnet werden.

$$\overline{D}^{E_i} = \frac{\overline{V}_i}{\overline{C}_i + \overline{A}_i} \quad . \tag{5.2-29}$$

Diese Korrektur der Einlagerungsdauer $\overline{D}^{E_i}$ hat bei den meist sehr geringen Fehlerwahrscheinlichkeiten q_i und den kleinen Zugriffswahrscheinlichkeiten π_i keine große Auswirkung, weil dann $\overline{C}_i$ um Größenordnungen größer als $\overline{A}_i$ ist. Somit gelten meist (5.2-5), (5.2-12b) und (5.2-22b) nahezu exakt.

Bei großen Fehlerwahrscheinlichkeiten und kleiner Anzahl von Auslagerungen sind jedoch diese Auswirkungen zu berücksichtigen. Dann ist $\overline{D}^{E_i}$ die Lösung des folgenden nichtlinearen Gleichungssystems, das iterativ gelöst werden kann.

Initialisierung: $A_i=0$

Iteration:

$$\overline{D}^{E_i} = \frac{\overline{V}_i}{\overline{C}_i + \overline{A}_i} \quad .$$

Bestimme q_i in Abhängigkeit von $\overline{D}^{E_i}$ mit (5.2-10), (5.2-16) und (5.2-18)!

Mit (5.2-29) kann $\overline{A}_i$ berechnet werden:

$$\overline{A}_i = a_i \cdot \sum_{\substack{1 \le j \le m \\ i \ne j}} q_j \cdot \overline{C}_j \quad .$$

Berechne $\overline{D}^{E_i}$ neu. Bei einem signifikanten Unterschied zum zuvor berechneten $\overline{D}^{E_i}$, iteriere noch einmal!

Die Itertion terminiert aus den folgenden Gründen:

Betrachtet man die Fehlerwahrscheinlichkeit q_i einer einzelnen Seite s_i in Abhängigkeit von D^{E_i}:

$q_i = f(D^{E_i})$,

dann ist f streng monoton wachsend mit D^{E_i}. Die Einlagerungsdauer D^{E_i} nimmt jedoch mit wachsendem q_i ab, d.h. die Funktion g:

$D^{E_i} = g(q_i)$

ist streng monoton fallend (vgl. Abbildung 5.11). Lösung des linearen Gleichungssystems ist der Schnittpunkt beider Funktionen.

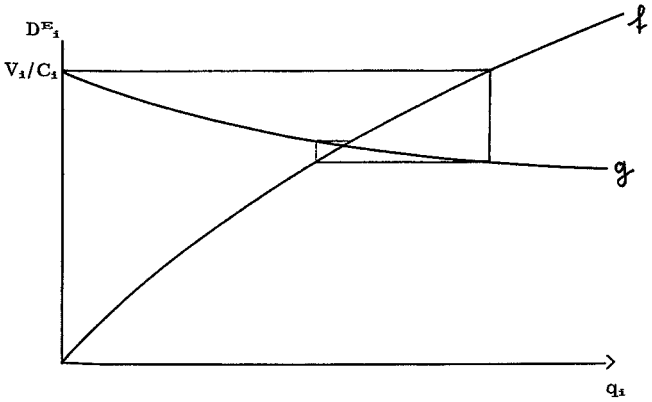


Abb. 5.11: **graphische Darstellung der Funktionen $q_i = f(D^{E_i})$ und $D^{E_i} = g(q_i)$**

5.3 Fehlerentdeckung beim Zugriff

In diesem Abschnitt wird vorausgesetzt, daß bei jedem **Zugriff** auf ein **Speicherwort** mit Hilfe von Coderedundanz eine Verfälschung entdeckt werden kann. Dies kann durch SEC-DED Codes, wie z.B. Hamming-Codes /SCHN 83/, geschehen. Für unser Modell gehen wir speziell von den folgenden **Voraussetzungen** aus.

- Wir betrachten ausschließlich **nicht maskierbare** Verfälschungen, weil nur diese nach außen hin Wirkung zeigen.

- Bei diesem Verfahren wird die Fehlerüberprüfung **wortweise** vorgenommen. Dies steht in gewissem Gegensatz zum Detaillierungsgrad unseres Modells, in dem nur Seiten- aber keine Wortzugriffe modelliert werden. Wir werden deshalb in diesem Abschnitt den Begriff "Zugriff" synonym mit **Seitenzugriff** verwenden. D.h. wir unterscheiden nicht zwischen einer Seitenreferenz und dem Zugriff auf ein bestimmtes Wort dieser Seite.

- Zusätzlich ergibt sich das Problem, daß verfälschte Speicherworte, auf die nicht mehr zugegriffen wird, unerkannt bleiben. Diese werden somit fehlerhaft auf den Hintergrundspeicher ausgelagert. Das Problem etwaiger Restfehler-Wahrscheinlichkeiten soll hier nicht weiter verfolgt werden.

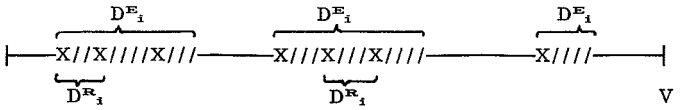
Im folgenden werden wieder ereignisunabhängige von ereignisabhängigen Störungen unterschieden.

Ereignisunabhängige Störungen

Wir wollen wiederum eine **einzelne Seite** s_i in **Phase I** betrachten. Bei jedem Zugriff auf s_i findet eine Fehlerüberprüfung statt, d.h. es gilt:

$C_i=Z_i$.

Der Zeitraum, in dem Störungen auftreten können, ist die Verweildauer von Seite s_i zwischen zwei Speicherzugriffen (vgl. Abbildung 5.12). Diese Zeitdauer wollen wir im weiteren **Inter-Referenz-Zeit** D^R_i nennen.



X: Überprüfung von Seite s_i beim Zugriff

Abb. 5.12: Fehlerentdeckung beim Zugriff

Wie im Modell aus Abschnitt 4.1 dargestellt, wird das Auftreten von ereignisunabhängigen Störungen durch einen allgemeinen (verzögerten) Erneuerungsprozeß modelliert (vgl. Abbildung 5.13).

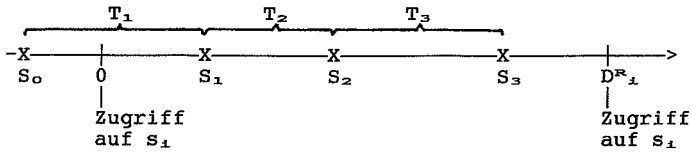


Abb 5.13: Ereignisunabhängiger Störungsprozeß für eine Fehlerentdeckung beim Zugriff

Berechnung der mittleren Inter-Referenz-Zeit

Die mittlere Inter-Referenz-Zeit $\overline{D}^{R_i}$ von s_i ist der Anteil eines Zugriffs an der Verweildauer dieser Seite. Finden $\overline{Z}_i$ Zugriffe statt, ergibt sich:

$$\overline{D}^{R_i} = \overline{V}_i / \overline{Z}_i$$

$$= \frac{1 - (1 - \pi_i)^T}{\pi_i} \cdot \frac{\overline{V}}{\beta \cdot B} \quad . \qquad (5.3\text{-}1)$$

Der zweite Faktor in (5.3-1) ist die mittlere Verweildauer zwischen zwei beliebigen Zugriffen. Sie wird mit einem von der Zugriffswahrscheinlichkeit abhängigen Faktor gewichtet.

Anzahl ereignisunabhängiger Störungen

Analog zum Vorgehen in Abschnitt 5.2 läßt sich nun die Verteilung der **Anzahl ereignisunabhängiger Störungen** N^U_i, die während der Inter-Referenz-Zeit aufgetreten sind, angeben. Mit Gleichung (4.1-5) ergibt sich:

$$\text{Prob}[N^U_i=k] = Pr^U(k,\bar{D}^R_i) \ . \qquad (5.3\text{-}2)$$

Den zugehörigen Mittelwert erhält man mit Hilfe des mittleren Erneuerungsabstandes (4.1-11):

$$\bar{N}^U_i = \bar{D}^R_i/\bar{s} = \frac{1-(1-\pi_i)^T}{\pi_i} \cdot \frac{\bar{V}}{\beta \cdot B} \cdot \bar{s}^{-1} \ . \qquad (5.3\text{-}3)$$

Beim Sonderfall exponential verteilter Störungsabstände ist die Anzahl der Störungen, wie in (4.1-14) angegeben, Poisson-verteilt.

Wie schon in (5.2-8) bis (5.2-10) argumentiert, kann mit Hilfe von (5.3-2), die **Wahrscheinlichkeit** q^U_i berechnet werden, mit der die Seite s_i von mindestens einer ereignisunabhängigen Störung betroffen ist, wenn jede Störung mit der Wahrscheinlichkeit $p_s=\bar{e}/H$ eine einzelne Seite betrifft.

$$q^U_i = \sum_{k=1}^{\infty} \{Pr^U(k,\bar{D}^R_i) \cdot \sum_{j=1}^{k} [\ (-1)^{j-1} \binom{k}{j} \cdot p_s^j\ ]\} \qquad (5.3\text{-}4)$$

Die **Wahrscheinlichkeit einer Fehlerentdeckung** q^U_i hängt (über die Größe $\bar{D}^R_i$, vgl. (5.3-1)) von der Verweildauer des Auftrags ab, die sich wiederum durch die von Fehlern induzierte Last erhöht. Diese rekursiven Abhängigkeiten und deren Modellierung werden im Kapitel 7 berücksichtigt.

Zugriffsinduzierte Störungen

Sehr viel leichter als im letzten Abschnitt läßt sich nun die Wahrscheinlichkeit berechnen, daß ein zugriffsinduzierter Speicherfehler aufgetreten ist. Bei jedem Zugriff wird nach Modellvoraussetzung mit der Wahrscheinlichkeit p_Z eine Störung verursacht. D.h. die Anzahl der bei den Zugriffen auf Seite s_i entdeckten Fehler N^Z_i ist wiederum binomisch verteilt und man erhält mit Gleichung (4.2-1):

$$\text{Prob}[N^Z_i= k] = Pr^Z(k,\pi_i \cdot \beta \cdot B) \ . \qquad (5.3\text{-}5)$$

Der Mittelwert ist dann:

$$\bar{N}^{Z}_{i} = p_Z \cdot \pi_i \cdot ß \cdot B \ . \qquad (5.3\text{-}6)$$

Die Wahrscheinlichkeit q^{Z}_{i}, mit der beim Zugriff auf s_i ein zugriffsinduzierter Fehler entdeckt wird, ist dann offensichtlich:

$$q^{Z}_{i} = p_Z \ . \qquad (5.3\text{-}7)$$

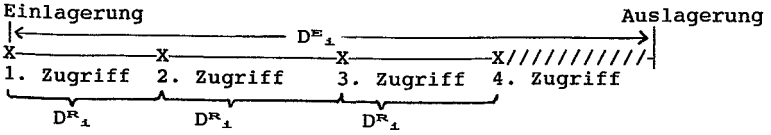
(5.3-7) ist gleichzeitig ein Sonderfall von Gleichung (5.2-16), mit $Z^{C}_{i}=1$.

Berechnung der mittleren Fehleranzahl in Seite s_i

Die oben berechneten Wahrscheinlichkeiten q^{U}_{i} und q^{Z}_{i} geben an, wie groß das Risiko ist, daß eine Seite s_i zwischen zwei Überprüfungen gestört wird. Die Wahrscheinlichkeit q_i, daß eine der beiden Störungsarten stattfindet, ist nach der Summenregel durch Gleichung (5.2-18):

$$q_i = q^{U}_{i} + q^{Z}_{i} - q^{U}_{i} \cdot q^{Z}_{i} \qquad (5.3\text{-}8)$$

gegeben. Mit Hilfe dieser Größe kann nun die mittlere Anzahl $\bar{N}^{F}_{i}$ der Fehler, die Seite s_i während ihrer Aufenthaltsdauer im Dispatching-System erleidet, errechnet werden.
Weil bei jedem Speicherzugriff eine Fehlerüberprüfung stattfindet, muß die Anzahl der Zugriffe auf Seite s_i mit der Wahrscheinlichkeit q_i einer Fehlerentdeckung gewichtet werden. Allerdings können nicht bei allen Zugriffen ereignisunabhänige Störungen aufgetreten sein:
Betrachtet man eine Seite s_i während ihrer Einlagerungsdauer, so erfolgt nach dem demand-paging-Prinzip /DESC 72/ eine Seiteneinlagerung nur dann, wenn auf eine nicht in der Arbeitsmenge vorhandene Seite zugegriffen wird. D.h. mit dem "ersten" Zugriff wird eine Seite in den Arbeitsspeicher eingelagert. Deshalb kann bei diesem ersten Zugriff auf eine Seite auch noch keine ereignisunabhängige Störung vorliegen, da ihr bisherige Verweildauer im Hauptspeicher 0 ist (vgl. Abbildung 5.14).



X: Zugriff auf s_i.

Abb. 5.14: Zugriffe auf die Seite s_i während ihrer Einlagerungsdauer

Ereignisunabhängige Störungen können also nicht bei allen Seitenzugriffen entdeckt werden, sondern nur dann, wenn die zugegriffene Seite schon einige Zeit im Hauptspeicher eingelagert war. Die "ersten" zu einer Einlagerung führenden Zugriffe müssen also unberücksichtigt bleiben. Die Anzahl $C_{i,2}$ der Überprüfungen, bei denen ereignisunabhängigen Fehler entdeckt werden können, ist also die Anzahl der Zugriffe auf s_i, die nicht mit einer Einlagerungen verbunden sind. Mit $\overline{C}_{i,1}$ aus Gleichung (5.2-6) ergibt sich:

$$\overline{C}_{i,2} = \overline{Z}_i - \overline{C}_{i,1}$$

$$= \pi_i \cdot [1 - (1-\pi_i)^T] \cdot \beta \cdot B \ . \qquad (5.3\text{-}9)$$

In (5.3-9) dienen die Indizes "1" und "2" zur Unterscheidung der verschiedenen Modelle. $\overline{C}_{i,1}$ bezieht sich auf das erste Verfahren der Fehlerentdeckung (beim Auslagern), $\overline{C}_{i,2}$ meint die entsprechende Größe für unser zweites in diesem Abschnitt beschriebenes Modell. Weil sich hier beide Modelle vermischen, wird diese Indizierung erforderlich. Alle übrigen Größen ohne diesen Index beziehen sich - wie in Abschnitt 5.1 vereinbart - auf das zweite Modell, d.h. auf eine Fehlerentdeckung beim Seitenzugriff.

Damit erhalten wir für unser Modell als **mittlere Fehleranzahl** $\overline{N}^F_i$ von Seite s_i:

$$\overline{N}^F_i = \overline{C}_{i,2} \cdot q_{i,2} + \overline{C}_{i,1} \cdot q^{z_{i,2}} \ . \qquad (5.3\text{-}10)$$

Im ersten Summand in (5.3-10) sind alle diejenigen Überprüfungen der Seite s_i enthalten, die sowohl ereignisunabhängige als auch zugriffsinduzierte Störungen feststellen können. Dies sind, wie eben gezeigt, $\overline{C}_{i,2}$ Zugriffe, die mit der Wahrscheinlichkeit $q_{i,2}$ gewichtet werden, mit der eine der beiden Fehler-Arten auftritt. Der zweite Summand gibt die Anzahl der Zugriffe an, bei denen ausschließlich zugriffsinduzierte Fehler entdeckt werden können. In ihm steckt die Anzahl $\overline{C}_{i,1}$ der ersten zu einer Einlagerung führenden Zugriffe auf s_i.

Mit (5.3-10) läßt sich die Rate Φ^F_i angeben, mit der bei Seite s_i Hauptspeicherfehler auftreten:

$$\Phi^F_i = \overline{N}^F_i / B \ . \qquad (5.3\text{-}11)$$

In (5.3-11) wurden wiederum die aufgetretenden Speicherfehler auf die Bearbeitungsdauer B bezogen.

Kritisch ist für eine Fehlerentdeckung beim Zugriff die Zeit zwischen dem letzten Seitenzugriff und der anschließenden Auslagerung der Seite (vgl. Abbildung 5.14). Tritt im genannten, in der Abbildung

schraffierten Zeitraum eine Störung auf, dann wird diese nicht mehr erkannt, weil kein Zugriff und damit auch keine Fehlerüberprüfung mehr stattfindet. Vielmehr wird die verfälschte Seite auf den Hintergrundspeicher zurückgeschrieben. Das Risiko einer solchen nicht entdeckten Störung soll nun bestimmt werden.

Nicht entdeckte Störungen

Nach insgesamt $\overline{C}_{i,1}$ Zugriffen, d.h. vor jeder Auslagerung, kann bei Seite s_i eine Störung auftreten, die nicht mehr entdeckt wird. Die Wahrscheinlichkeit, daß nach dem "letzten" Zugriff auf s_i noch eine Störung auftritt, läßt sich nach (5.3-4) durch $q^U_{i,2}$ abschätzen, wenn man zur Vereinfachung annimmt, daß die Interreferenzzeit vor einer Auslagerung mit dem selben Mittelwert verteilt ist, wie die vorhergehenden. Danach ist die **Anzahl nicht entdeckter Störungen** $\overline{F}^{ne}_i$ für Seite s_i:

$$\overline{F}^{ne}_i = \overline{C}_{i,1} \cdot q^U_{i,2} \quad . \qquad (5.3\text{-}12)$$

Eine Möglichkeit zur Verhinderung nicht entdeckter Fehler ist in Speicherverwaltungen mit "Durchschreiben" gegeben.

Speicherverwaltungssysteme mit Durchschreiben

In diesen Systemen wird jede Seiten-Änderung im Hauptspeicher augenblicklich auf den Hintergrundspeicher "durchgeschrieben". D.h. ein vollständiges und mit einer Überprüfung verbundenes Rückschreiben der Seite beim Auslagern muß nicht durchgeführt werden. Es gibt also keine mit einer Übertragung in den Hintergrundspeicher verbundene Auslagerung, sondern nur die logische Freigabe der entsprechenden Seite im Arbeitsspeicher. Es wird u.a. auch deswegen so vorgegangen, um Seiten, die während einer Auftragsbearbeitung nicht verändert worden sind, ohne explizites Rückschreiben auf den Hintergrundspeicher auslagern zu können.
Wird jede Seitenänderung sofort "durchgeschrieben", führen demnach Störungen zwischen dem letzten Zugriff und einer Auslagerung zu keinem Fehler.

Durch Speicherfehler induzierte Last

Werden Speicherfehler bereits beim Zugriff entdeckt, ist die Fehlerbehandlung denkbar einfach. Bevor die Daten der betroffenen Seite benutzt werden, wird der Fehler bereits festgestellt. Daher kann eine Ausbreitung (error propagation) des Fehlers auf andere Speicherseiten **ausgeschlossen** werden. Deshalb ist als einzige Maßnahme ein

erneutes Einlagern der betroffenen Seite erforderlich. Bei diesem Verfahren ist also **kein rollback** notwendig und die Ausführungsdauer wird somit nicht erhöht, d.h. es gilt:

$$B^{R}_{i} = 0 \quad , \; \Gamma^{C}_{i} = 1 \; . \tag{5.3-13}$$

Die Auswirkungen von Fehlern auf die Systemleistung beschränken sich demnach auf eine

- Erhöhung der Paging-Rate, und eine damit verbundene
- größere Verweildauer des Auftrags.

Auch hier sind wieder rekursive Abhängigkeiten festzustellen: Die durch Fehler erhöht Paging-Rate verlängert unmittelbar die Verweildauer des Auftrags.

Erhöhung der Paging-Rate

Insgesamt werden während der Durchführung des gesamten Auftrags

$$\overline{\Gamma}^{R} = \sum_{1 \le i \le m} \overline{N}^{F}_{i} \tag{5.3-14}$$

Fehler beim Zugriff entdeckt.
Analog zur Gleichung (5.2-28) ergibt sich für die zusätzliche Paging-Rate aufgrund von Hauptspeicherfehlern im zweiten Modell

$$\Phi^{HF} = \overline{\Gamma}^{R}/B \; . \tag{5.3-15}$$

5.4 Fehlerentdeckung durch Konsistenzüberprüfung

In diesem Abschnitt wird vorausgesetzt, daß Fehlerüberprüfungen nach einer bestimmten, festen Bearbeitungszeit durchgeführt werden. Dabei sind die folgenden Vorgehensweisen möglich:

- Die Fehlerüberprüfung kann mit Hilfe von **Software** durch spezielle Akzeptanztests bzw. Konsistenzchecks, z.B. durch Diagnose-Routinen, /RAND 75/ geschehen. Eine Darstellung verschiedener Prüfalgorithmen zum Testen von Speichern wird in /HUNG 82/ gegeben. Entsprechende Diagnose-Routinen werden in festen Zeitabständen beispielsweise vom Betriebssystem eines VAX-Rechners angestoßen /SIEW 86/. Die Zeitpunkte einer solchen Überprüfung werden auch **checkpoints** /RAND 75/ genannt. Einen Überblick über die Problematik von Akzeptanztests durch Software bietet /CLSK 87/.

- In hochzuverlässigen, fehlertoleranten Rechnern werden oft Konsistenzchecks per **Hardware** durchgeführt. Fehler werden dabei durch **Votierung** bzw. Vergleich der Ergebnisse mehrfach vorhandener identischer Aufträge entdeckt bzw. maskiert /PRAD 86/. Dabei werden die Akzeptanztests oft schon vor Ende eines Auftrags, wenn z.B. Zwischenergebnisse vorliegen, durchgeführt. Bei einem Votieren der Auftragsergebnisse können nicht nur ausschließlich Speicherfehler, sondern auch andere Fehler erkannt werden.

Wir wollen in diesem Abschnitt **voraussetzen**, daß die genannten Verfahren der folgenden Art sind:

- Ein Testen des Auftrags erfolgt immer genau dann, wenn er eine Bedienzeit von B^K erhalten hat. D.h. der Testzeitpunkt korrespondiert nicht mit einem bestimmten Ereignis der Speicherverwaltung oder der Auftragsbearbeitung. (Ein Sonderfall ist dabei $B^K = B$, d.h. ein Test am Ende der Bedienzeit).

- Wird nach der festen Bedienzeit B^K ein Speichertest durchgeführt, so soll dieser sämtliche während B^K vom Auftrag referenzierten Seiten überprüfen.

- Fehler, die wegen der im Verfahren gegebenen Restfehler-Wahrscheinlichkeit nicht erkannt werden, wollen wir nicht betrachten.

Um bei diesem Vorgehen eine Fehlerbehebung gewährleisten zu können, muß das Betriebssystem im Fehlerfall den Auftrag ab dem letzten

checkpoint wieder neu starten und seine Datenintegrität wiederherstellen können. Dazu muß an jedem checkpoint der gesamte Auftrags-Kontext, einschließlich aller benutzten Speicherseiten, gesichert werden. Kann das Betriebssystem diese Forderung nicht erfüllen, muß im Fehlerfall der gesamte Auftrag wiederholt werden.
Ein checkpointing verursacht natürlich einen entsprechend großen overhead, es läßt sich dafür aber auch weitgehend durch Software realisieren. Das Konzept eines software-implementierten checkpointings wurde z.B beim SIFT-Rechner /WENS 78/ eingesetzt. Ein rollback mit Hilfe eines Cache-Speichers wird in /HUMA 87/ vorgestellt.

Zunächst betrachten wir nur die Fehler bei einem einzelnen Auftrag, d.h. etwaige Auftragskopien bei votierenden Systemen bleiben vorerst unberücksichtigt. Als erstes werden wieder ereignisunabhängige Fehler untersucht.

Ereignisunabhängige Fehler

Genau wie in den beiden vorhergehenden Modellen ist die Verweildauer zwischen zwei Fehlerüberprüfungen der Zeitraum, in dem Störungen auftreten können. Diese Zeit soll im weiteren **Inter-Konsistenzcheck-Zeit** D^K genannt werden. Im Gegensatz zu den Verfahren der vorhergehenden Abschnitte werden implizit sämtliche vom Auftrag benutzten Speicherseiten überprüft. Oft wird gleichzeitig auch die korrekte Ausführung des Auftrags seit dem letzten checkpoint kontrolliert.

Wird laut Voraussetzung nach jedem festen Bearbeitungsabschnitt B^K eine Fehlerüberprüfung durchgeführt, dann gilt für die mittlere Anzahl C der Konsistenzprüfungen

$$\bar{C} = B/B^K \quad . \tag{5.4-1}$$

Wird nach jedem Bearbeitungsabschnitt der Dauer B^K eine Fehlerüberprüfung durchgeführt, dann ist die zugehörige mittlere Inter-Konsistenzcheck-Zeit D^K der relative Anteil an der Gesamtverweildauer V, und man erhält:

$$\bar{D}^K = \frac{\bar{V}}{\bar{C}} = \frac{B^K}{B} \cdot \bar{V} \quad . \tag{5.4-2}$$

Ereignisunabhängige Störungen lassen sich wiederum durch einen verzögerten Erneuerungsprozeß modellieren. Wie bei den anderen Verfahren folgt mit Gleichung (4.1-5) die Verteilung für die Anzahl N^U ereignisunabhängiger Störungen im Hauptspeicher:

$$\text{Prob}[N^U=k] = Pr^U(k,\bar{D}^K) \quad . \tag{5.4-3}$$

Die mittlere Anzahl $\bar{N}^U$ der Störungen während dieser Zeit ist dann:

$$\bar{N}^U = \frac{\bar{V}}{C} \cdot \bar{s}^{-1} . \qquad (5.4\text{-}4)$$

In den Gleichungen (5.4-1) bis (5.4-4) fehlt der Index i, weil sich die Fehlerprüfung nicht auf eine, sondern auf alle während der Inter-Konsistenzcheck-Zeit vom Auftrag benutzten Seiten bezieht.

Uns interessiert somit nicht die Wahrscheinlichkeit, mit der eine bestimmte virtuelle Seite s_i gestört wird, sondern ob eine Verfälschung bei mindestens einer Seite aus der Arbeitsmenge vorliegt.
Nach unserem Modell aus Abschnitt 4.1.2 sind von einer Störung im Mittel $\bar{e}$ Seiten betroffen, sodaß eine einzelne Seite mit der Wahrscheinlichkeit $\bar{e}/H$ verfälscht wird. Enthält die Arbeitsmenge w(T) Speicherseiten, dann errechnet sich die Wahrscheinlichkeit p_a mit der eine dieser Seiten von einer Störung beroffen ist durch die Formel von Poincaré-Sylvester:

$$p_a | w(T)=k = \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{j} \cdot \left(\frac{\bar{e}}{H}\right)^j .$$

Ist durch g(k,T):=Prob[w(T)=k] die Verteilung der Arbeitsmengengröße gegeben, dann erhält man nach dem Satz von der totalen Wahrscheinlichkeit

$$p_a = \sum_{k=1}^{T} g(k,T) \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{j} \cdot \left(\frac{\bar{e}}{H}\right)^j . \qquad (5.4\text{-}5)$$

Die Wahrscheinlichkeit q^U, mit der bei der Konsistenzprüfung eine ereignisunabhängige Störung entdeckt wird, erhält man analog zu den vorhergehenden Abschnitten durch eine erneute Anwendung der Formel von Poincaré-Sylvester:

$$q^U = \sum_{k=1}^{\infty} Pr^U(k,\bar{D}^K) \sum_{j=1}^{k} (-1)^{j-1} \binom{k}{j} \cdot p_a^j . \qquad (5.4\text{-}6)$$

Auch in diesem Modell muß auf die Rekursivität zwischen Fehleraufttritts-Wahrscheinlichkeit q^U und der Verweildauer D^K zwischen zwei Fehlerüberprüfungen hingewiesen werden.

Zugriffsinduzierte Fehler

Um die Wahrscheinlichkeit eines zugiffsinduzierten Fehlers zu berechnen, muß zunächst die mittlere Anzahl $\bar{Z}^C$ der Zugriffe zwischen zwei Fehlerüberprüfungen bestimt werden. Sie ist

$$\bar{Z}^C = B^K \cdot ß . \qquad (5.4\text{-}7)$$

Die **Anzahl der zugriffsinduzierten Fehler** N^Z ist dann nach unserem Modell aus Kapitel 4 wieder binomisch verteilt:

$$Prob[N^Z=k] = Pr^Z(B^K \cdot ß, k) \ , \qquad (5.4\text{-}8)$$

siehe (4.2-1). Die Wahrscheinlichkeit q^Z, daß (mindestens) ein zugriffsinduzierter Fehler beim Akzeptanztest gefunden wird, ist 1 minus der Wahrscheinlichkeit, daß alle $\overline{Z}^C$ Zugriffe keinen Fehler verursacht haben. Man erhält somit in Analogie zur Gleichung (5.2-18):

$$q^Z = 1 - (1 - p_z)^{B^K \cdot ß} . \qquad (5.4\text{-}9)$$

Berechnung der mittleren Anzahl Fehlerentdeckungen:

Die oben berechnete Wahrscheinlichkeiten q^U und q^Z geben an, wie groß das Risiko ist, daß die während B^K vom Auftrag in den Hauptspeicher eingelagerten Seiten zwischen zwei Überprüfungen gestört worden sind. Die Wahrscheinlichkeit q, daß eine der beiden Fehlerarten während $\overline{D}^K$ aufgetreten ist, ergibt sich wieder durch die Summenformel zu:

$$q = q^U + q^Z - q^U \cdot q^Z \ . \qquad (5.4\text{-}10)$$

Mit Hilfe der Größe q kann die mittlere Anzahl der Fehlerentdeckungen N^F, die einen bestimmten Auftrag während seiner Aufenthaltsdauer im Dispatching-System betreffen, errechnet werden:

$$\overline{N^F} = q \cdot \overline{C}$$

$$= q \cdot (B/B^K) \ . \qquad (5.4\text{-}11)$$

Exkurs: Votierende Systeme

Bei votierenden Systemen werden Konsistenüberprüfungen durch den Einsatz massiver Hardware- bzw. Zeit-Redundanz erreicht: Es gibt identische Kopien eines Auftrags, die von diversitärer Hardware bearbeitet werden. Bei der Untersuchung von Speicherfehlern muß bei diesem Verfahren berücksichtigt werden, daß offensichtlich ein Vielfaches des Speicherplatzes belegt wird, und sich dadurch das Risiko eines Speicherfehlers erhöht.

Die Gleichung (5.4-10) gibt die Wahrscheinlichkeit an, mit der bei der Konsistenzüberprüfung einer einzelnen Auftragskopie ein Fehler bemerkt wird.
Im folgenden untersuchen wir Fehler bei einem aus mehreren Auftragskopien bestehenden Auftrag. Dabei wollen wir annehmen, daß jede der

korrespondierenden Auftragskopien mit derselben Wahrscheinlichkeit gestört wird.

In Abhängigkeit von der Anzahl der identischen Auftragskopien wollen wir zwei Fälle unterscheiden:

1. Fall: **Es gibt zwei Auftragskopien.**
In diesem Fall ist die Konsistenzüberprüfung nichts anderes als ein Vergleich der beiden parallel oder zeitverschoben ablaufenden Aufträge. Hier kann nur eine Fehlerentdeckung und keine Fehlermaskierung stattfinden. In unserem Modell wollen wir davon ausgehen, daß die Störungen in den Arbeitsmengen beider Aufträge stochastisch unabhängig sind, was bei einer Auftragsausführung auf diversitärer Hardware durchaus vorausgesetzt werden kann. Außerdem nehmen wir an, daß Fehler auch erkannt werden, wenn sie in beiden Auftragskopien auftreten. Diese Annahme impliziert, daß zwei Störungen normalerweise nicht beide Aufträge an gleicher Stelle in derselben Art und Weise verfälschen (common mode failures).
Die Wahrscheinlichkeit einer Fehlerentdeckung q^{D2}, d.h. mit der genau einer der beiden korrespondierenden Aufträge verfälscht wird, ist binomisch verteilt und wir erhalten:

$$q^{D2} = 2q \cdot (1-q)$$

$$= 2q - 2q^2 \; . \qquad (5.4\text{-}12)$$

2. Fall: **Es gibt drei Auftragskopien.**
Diese Anzahl von Auftragskopien ermöglicht eine **2-von-3-Mehrheitsentscheidung**. Man kann somit auch Fehler maskieren, solange nur ein einziger Auftrag einen Fehler erleidet. Die Wahrscheinlichkeit q^{M3}, daß ein Fehler auftritt und maskiert wird, ist gegeben durch:

$$q^{M3} = \binom{3}{1} q \cdot (1-q)^2$$

$$= 3 \cdot q - 6 \cdot q^2 + 3 \cdot q^3 \; . \qquad (5.4\text{-}13)$$

Die Wahrscheinlichkeit q^{D3} eines entdeckten und nicht maskierbaren Fehlers entspricht der Auftrittswahrscheinlichkeit von zwei oder drei verfälschten Auftragskopien.

$$q^{D3} = \binom{3}{2} q^2 \cdot (1-q) + \binom{3}{3} q^3$$

$$= 3 \cdot q^2 - 2 \cdot q^3 \; . \qquad (5.4\text{-}14)$$

Die Gleichungen (5.4-12) bis (5.4-14) können leicht auf Fälle mit mehr als drei Auftragskopien mit der Formel für die Binominal-Verteilung verallgemeinert werden.

Bei **votierenden Systemen** muß q in Gleichung (5.4-10) durch die Wahrscheinlichkeiten q^{D2} bzw. q^{D3} ersetzt werden.

Durch Speicherfehler induzierte Last

Bei einer Fehlerentdeckung weiß man hier im Gegensatz zu den vorhergehenden Verfahren weder welche, noch wieviele der referenzierten Seiten verfälscht wurden. Somit müssen alle nach der letzten Konsistenzüberprüfung referenzierten Seiten neu eingelagert werden.

Die zur Fehlerbehandlung durchzuführenden Maßnahmen sind denen bei einer Fehlerentdeckung beim Seitenauslagern sehr ähnlich:

- Es ist ein rollback des betroffenen Auftrags bis zum Zeitpunkt des letzten Konsistenzchecks erforderlich.
- Alle während des Bearbeitungsabschnitts benutzten Seiten müssen zur Wiederherstellung der Datenintegrität neu eingelagert werden.

Die durch diese Behandlungsmaßnahmen induzierte Last wird durch

- eine Verlängerung der Bedienzeit des Auftrags B und
- eine Erhöhung der Paging-Rate

quantifiziert.

Verlängerung der Bedienzeit B

Bei jedem entdeckten Fehler muß der gesamte letzte Bedienabschnitt, d.h. eine Bediendauer von B^K wiederholt werden. Es gilt offensichtlich:

$$B^C = B^K \quad , \tag{5.4-15}$$

wenn B^C die zu wiederholende Bediendauer ist, um einen einzelnen, bei einer Konsistenzüberprüfung entdeckten Fehler zu behandeln.

Nach (5.4-11) finden im Mittel $\overline{N}^F$ Fehlerentdeckungen statt, so daß die Bedienzeit B des Auftrags insgesamt um die mittlere Zeitdauer

$$\overline{B}^R = \overline{N}^F{}_J \cdot B^K$$

$$= q \cdot B \qquad (5.4\text{-}16)$$

vergrößert wird.

Erhöhung der Paging-Rate

Wie in Abschnitt 5.2 muß die mittlere Anzahl aller verschiedenen Seiten bestimmt werden, auf die während B^K zugegriffen wurde, denn genau dies ist die mittlere Anzahl der erneut einzulagernden Seiten.
Die Anzahl $\overline{\Gamma}^C$ der zur Wiederherstellung der Datenintegrität erforderlichen Seiteneinlagerungen bestimmen wir genau wie bei der Herleitung von Gleichung (5.2-26) :

$$\overline{\Gamma}^C = \sum_{1 \le j \le m} \{1 - (1 - \pi_j)^{\beta \cdot B^K}\}$$

$$= m - \sum_{1 \le j \le m} (1 - \pi_j)^{\beta \cdot B^K} . \qquad (5.4\text{-}17)$$

Der Exponent in Gleichung (5.4-17) ist dabei die Anzahl der Zugriffe auf alle Seiten, die in einem Bearbeitungsabschnitt erfolgen.
Mit diesem Ergebnis kann die mittlere Anzahl Γ^R aller durch Fehler verursachten Einlagerungen bestimmt werden:

$$\overline{\Gamma}^R = \overline{N}^F \cdot \overline{\Gamma}^C . \qquad (5.4\text{-}18)$$

Die entsprechende Rate ist dann

$$\Phi^{HF} = \overline{\Gamma}^R / B . \qquad (5.4\text{-}19)$$

5.5 Fehlerentdeckung bei der Seiteneinlagerung

Zur Untersuchung von Fehlern bei der Seiteneinlagerung muß zunächst die Gesamtzahl aller Seiteneinlagerungen bestimmt werden. Anschließend kann das Störungsmodell aus Abschnitt 4.2.2 angewendet werden.

Anzahl aller Seitenwechsel

Die mittlere Anzahl $\overline{\Gamma}^{P}$ aller Seitenwechsel, die während der fehlerfreien Auftragsausführung aufgrund von Paging stattfinden, ist gleich der Anzahl aller Seiteneinlagerungen. Es ergibt sich also mit (5.2-4):

$$\overline{\Gamma}^{P} = \sum_{1 \le i \le m} \overline{C}_{i,1}$$

$$= \sum_{1 \le i \le m} [\pi_i (1-\pi_i)^{T} \cdot \beta + (1-(1-\pi_i)^{T})] \ . \qquad (5.5\text{-}1)$$

Nach den Überlegungen der vorangegangenen Abschnitte sind zur Behandlung aufgetretener Fehler weitere Seiteneinlagerungen erforderlich.

Sei $q_{i,j}$ die für das Verfahren j errechnete Wahrscheinlichkeit mit der bei einer Fehlerüberprüfung ein Fehler entdeckt wird. (Dabei verweist im weiteren der Index j, mit j=1,2,3, auf das entsprechende, in Abschnitt 5.1 definierte Modell.)
Dann ist die mittlere Anzahl $\overline{N}^{F}_{j}$ der bei Verfahren j entdeckten Fehler durch die Gleichungen (5.2-23), (5.3-10), bzw. (5.4-11) gegeben.

Zur Wiederherstellung der Datenintegrität sind zusätzliche Einlagerungen erforderlich. Ihre Anzahl $\overline{\Gamma}^{R}_{j}$ ergibt sich für die ersten beiden Modelle gemäß (5.2-26) und (5.3-14) zu:

$$\overline{\Gamma}^{R}_{j} = \sum_{1 \le i \le m} \overline{N}^{F}_{i,j} \cdot \overline{\Gamma}^{C}_{i,j} \ , \qquad j \varepsilon \{1,2\} \ , \qquad (5.5\text{-}2)$$

wobei $\overline{\Gamma}^{C}_{i,j}$ durch (5.2-25) bzw. (5.3-13) gegeben ist.
Beim dritten Modell, der Fehlerentdeckung durch Konsistenzüberprüfung, werden nach Voraussetzung alle während B^{K} referenzierten Seiten getestet, so daß eine Summation über alle virtuellen Seiten entfällt. Es gilt dann (5.4-18):

$$\overline{\Gamma}^{R}_{3} = \overline{N}^{F}_{3} \cdot \overline{\Gamma}^{C}_{3} \ . \qquad (5.5\text{-}3)$$

Nach (5.5-1) bis (5.5-3) müssen insgesamt

$$\bar{\Gamma}_j = \bar{\Gamma}^{P} + \bar{\Gamma}^{R}_j \qquad (5.5\text{-}4)$$

Seiten eingelagert werden. Der erste Summand verweist auf die im fehlerfreien Betrieb erforderlichen Seitenwechsel, der zweite Summand umfaßt alle zur Fehlerbehandlung erforderlichen Seiteneinlagerungen.

Anzahl der beim Seiteneinlagern entdeckten Fehler

Nach Abschnitt 4.2.2 induziert jeder Seitenwechsel mit der Wahrscheinlichkeit p_P einen Fehler. Mit (4.2-3) ist dann für das Verfahren jc{1,2,3} die **Anzahl N^P_j der Paging-Fehler** ebenfalls binomisch verteilt und man erhält bei $\bar{\Gamma}_j$ Seitenwechseln:

$$\begin{aligned} \text{Prob}[N^P_j=k] &= Pr^P(k,\bar{\Gamma}_j) \ , \\ &= \binom{\bar{\Gamma}_j}{k} \cdot p_P^{\,k} \cdot (1-p_P)^{\bar{\Gamma}_j-k} , \end{aligned} \qquad (5.5\text{-}5)$$

bzw. den Mittelwert

$$\bar{N}^P_j = \bar{\Gamma}_j \cdot p_P \ . \qquad (5.5\text{-}6)$$

Behandlung von Paging-Fehlern

Fehler beim Seitenwechsel werden im allgeinen sofort bemerkt, d.h. sie können sich nicht auf andere Speicherseiten auswirken. Als Behandlungsmaßnahme reicht deshalb ein erneutes Einlagern der betroffenen Seite aus. D.h. $\bar{N}^P$ quantifiziert also auch den Umfang der durch Paging-Fehler verursachten Seiteneinlagerungen.
Allerdings können auch diese $\bar{N}^P$ Seitenwechsel wiederum von Fehlern betroffen werden, sodaß für die **Gesamtzahl aller Paging-Fehler $\bar{N}^{P*}$** die Gleichung (5.5-6) modifiziert werden muß. Es ergibt sich mit der Formel für geometrische Reihen für das Verfahren j:

$$\bar{N}^{P*}_j = \bar{\Gamma}_j \cdot \frac{p_P}{1 - p_P} \ . \qquad (5.5\text{-}7)$$

Durch (5.5-7) ist gleichzeitig die durch Paging-Fehler induzierte Last beschrieben. Als entsprechende Rate Φ^{PF}_j aller Paging-Fehler erhält man:

$$\Phi^{PF}_j = \bar{N}^{P*}_j / B \ . \qquad (5.5\text{-}8)$$

6 Approximatives Warteschlangen-Modell zur Leistungsuntersuchung

In diesem Kapitel soll das **Leistungsverhalten** von Systemen mit virtueller Speicherverwaltung untersucht werden. Dies ist aus zwei Gründen erforderlich. Erstens sollen die Auswirkungen von Fehlern auf typische Leistungsindizes, wie Durchsatz und Antwortzeiten untersucht werden. Zweitens gehen in unser Fehlermodell Leistungsgrößen, wie z.B. die Verweildauer eines Auftrags im Hauptspeicher, ein. Wie sich diese durch ein Warteschlangenmodell bestimmen lassen, ist Gegenstand dieses Kapitels.
Weil Fehler im Vergleich zu anderen Ereignissen im Dispatching-System um Größenordnungen seltener auftreten, reicht es, das **stationäre** Leistungsverhalten zu betrachten /MEYE 82/. In unserem Modell wollen wir in besonderem Maße das **Speicherverhalten** von Aufträgen berücksichtigen, wobei insbesondere das **Phasenverhalten** /DEKA 75/ der Aufträge eingehen soll. Wie wir sehen werden, erschwert dieser Wunsch allerdings die Modellierung mit Hilfe von Warteschlangen-Netzwerken (WS-Netze), weil dann **keine exakte analytische (Produktform-) Lösung** vorliegt.
Die größte Klasse der Warteschlangen-Netze, für die es eine exakte analytische (Produktform-)Lösung gibt, sind die BCMP-Netze /BCMP 75/. Die Voraussetzungen und Einschränkungen dieser Klasse werden jedoch in unserem Modell in einigen wesentlichen Punkten verletzt, sodaß nur eine **approximative Lösung** möglich ist.

Verletzung der BCMP-Voraussetzungen

- Mit der sich dynamisch ändernden Arbeitsmengengröße und dem zeitabhängigen Lokalitätsverhalten variiert die Paging-Rate eines Auftrags. Denn die Dauer einer CPU-Belegung und die Wahrscheinlichkeit, mit der nach einer CPU-Belegung ein Paging erfolgt, ist nicht zeitkonstant oder stationär, wie im BCMP-Modell vorausgesetzt, sondern sie verändern sich mit dem Speicherverhalten des Auftrags.

- Ähnliches gilt auch für das Fehlerverhalten des Systems. Die Auftritts-Wahrscheinlichkeiten/-Raten von Speicher- und Paging-Fehlern sind nicht nur auftragsspezifisch, sondern hängen auch von Arbeitsmengengrößen und Verweildauern ab und sind somit ebenfalls zeitabhängig.

- Die Last im Dispatching-System ist nicht konstant. Nach dem working-set-Prinzip /DENN 68/ wird ein Auftrag genau dann in den Hauptspeicher eingelagert, wenn für seine Arbeitsmengen-

größe genügend Speicherplatz zur Verfügung steht. Die Anzahl der Aufträge im Dispatching-System sowie die Phasen, in denen sie sich aktuell befinden, ändern sich damit während der Betriebszeit.

Insgesamt sind die BCMP-Voraussetzungen also nicht erfüllt, denn es können weder ein geschlossenes Warteschlangen-Netz mit konstanter Auftragsanzahl noch zeitunabhängige Bedienraten und Übergangswahrscheinlichkeiten angenommen werden.
Damit ist eine **exakte** analytische Lösung mit Hilfe von BCMP-Modellen nicht möglich. Ein in der Literatur zur Leistungsbewertung /BARD 77/, /BRBC 77/, /BRAN 75/, /COVA 76/, /HEMO 81/, /LAVE 83/ erfolgreich eingeschlagener, und auch in dieser Arbeit verfolgter Weg ist die **Dekompositions-Approximation** /COUR 77/, /GEMI 80/ nach P.J. Courtois. In dieser Arbeit werden die Leistungsmodelle allerdings so erweitert, daß sie auch das Auftreten von Fehlern berücksichtigen.

Zur Bestimmung der relevanten Leistungsindizes wird nach dem Ansatz von Brandwajn /BRAN 85/ ein approximatives Modell in zwei Schritten entwickelt:
Zuerst werden die **marginalen Zustandswahrscheinlichkeiten** bestimmt, d.h. hier die Wahrscheinlichkeiten, mit denen sich eine bestimmte Last im Dispatching-System befindet.
Unter der Voraussetzung, daß sich im Dispatching-System eine bestimmte Last befindet, können anschließend die **bedingten** stationären **Zustandswahrscheinlichkeiten** berechnet werden, mit deren Hilfe sich Leistungsmaße ableiten lassen. In Abschnitt 6.2 wird dazu ein WS-Netz spezifiziert, das mit der **mean value anlysis** /REIS 79/, /RELA 80/ gelöst wird.

6.1 Modell der Systemlast

In diesem Abschnitt soll ein Modell für die Last im Dispatching-System entwickelt werden. Dieses Last-Modell spielt eine wichtige Rolle in unserem Gesamtmodell. Denn die im System vorhandene Last, d.h. die Anzahl der speicherresidenten Aufträge und ihr spezielles Lastprofil, beeinflußt in entscheidender Weise die für unser Fehler-Modell benötigte Paging-Rate und die Verweildauer von Aufträgen.

Im **realen System** wird die Anzahl der sich im Hauptspeicher befindenden Aufträge durch den Scheduler des Betriebssystems bestimmt. Dieser teilt die vom System zu bearbeitenden Aufträge gewöhnlich in verschiedene Auftragsklassen ein, und führt umfangreiche Analysen durch, um einen geeigneten **job-mix** zu erstellen. Solch ein geeigneter job-mix enthält aus jeder Auftragsklasse gerade soviele Aufträge, daß hohe Betriebsmittel-Auslastungen und hohe Durchsätze mit akzeptablen Antwortzeiten verbunden sind /BRIN 77/.

Um das Lastverhalten unterschiedlicher Auftragsklassen modellieren zu können, werden in Warteschlangen-**Modellen** entsprechend viele verschiedene **Lastklassen** /LAVE 83/ angenommen. Jede Lastklasse R weist unterschiedliche Charakteristika auf, z.B.

- CPU-Bedienzeitanforderung B_R,
- Paging-Rate $\Phi^P{}_R$ und
- Arbeitsmengengröße $w_R(T)$.

Zunächst stellt sich die Frage, welche Lastklassen es in unserem Modell geben soll. Weil es bei unserer Problemstellung um das Speicherverhalten geht, wollen wir die Aufträge ausschließlich anhand ihres Speicherbedarfs differenzieren.

Wie in Kapitel 2 durch (2.1-8) spezifiziert, durchläuft ein Auftrag L Phasen, in denen er jeweils unterschiedlichen Speicherbedarf und verschiedene Paging-Raten aufweist. Bei dieser Modellsicht entspricht jeder Auftragsphase in unserem Modell eine Klasse im Warteschlangen-Netz, d.h. in unserem Modell gibt es für jede Auftragsphase eine eigene Lastklasse. Der stationäre **Zustandsvektor K** ist also durch das L-Tupel

$$\mathbf{K} = (k_1, k_2, \ldots, k_L) \quad \text{, mit} \qquad (6.1\text{-}1)$$

k_I := Anzahl der Aufträge aus Klasse I für I=1,..,L .

gegeben.

Besitzt der Hauptspeicher eine Größe von H Seiten und hat jede Lastklasse I einen mittleren Speicherbedarf von $\overline{w}_I(T)$, so werden nach dem working-set-Prinzip /DENN 68/ nur soviele Aufträge aktiviert, daß deren Arbeitsmengen in den Hauptspeicher hineinpassen. D.h. im Modell muß die **Randbedingung** gelten:

$$\sum_{1 \le I \le L} k_I \cdot \overline{w}_I(T) \le H \ . \qquad (6.1\text{-}2)$$

Bemerkung
In Gleichung (6.1-2) betrachten wir nur den Mittelwert der Arbeitsmengengröße. Dies ist als (vereinfachtes) Modell der Dispatching-Strategie gerechtfertigt, weil der Dispatcher den Speicher oft nach dem geschätzten voraussichtlichen Speicherbedarf eines Auftrags zuteilt.

Ziel unseres Lastmodells ist u.a., die **Zustands-Verteilung** des Vektors **K** unter Berücksichtigung der Randbedingung (6.1-2) zu bestimmen. Dabei sind zwei verschiedene Wege gangbar:

- Zum einen läßt sich bei einem gegebenen System die Zustands-Verteilung von **K** ohne eine spezielle Modellentwicklung durch **Messungen** /BEIL 88/ ermitteln. Mit Hilfe von Software- oder Hardware-Monitoren muß dann gemessen werden, wie sich die Anzahl der speicherresidenten Aufträge und deren Arbeitsmengengrößen verteilen.

- Ein anderer, aufwendigerer Weg ist die Herleitung eines **stochastischen Last-Modells**, das im folgenden skizziert werden soll:
  Bei unserer bisherigen Modell-Betrachtung sind wir stets von den Vorgängen im Dispatching-System ausgegangen. Auch hier wollen wir nicht das Benutzer-Verhalten und spezielle Scheduling-Strategien im "äußeren System", d.h. den Auftragsfluß außerhalb des Dispatching-Systems, modellieren. Dies würde die Fragestellung dieser Arbeit überschreiten. Prinzipiell bereitet eine solche Modellierung jedoch keine größeren Schwierigkeiten. Diesbezüglich sei auf die Literatur /BRBC 77/, /SCHO 76/, /BARD 77/, /LAZO 84/, /HEMO 81/ verwiesen.

Wir wollen vielmehr **voraussetzen**, daß

- sich insgesamt K Aufträge im Dispatching-System befinden, wobei

  $K = \sum_{1 \le I \le L} k_I$ vorgegeben ist.

(Die Verteilung von K wird als gegeben angenommen.)

Dann interessiert uns, mit welcher Wahrscheinlichkeit sich die K Aufträge in den verschiedenen Phasen befinden. Gesucht ist die Auftritts-Wahrscheinlichkeit eines Lastvektors $\mathbf{K}$

$$P_{k_1,k_L} := \text{Prob}[\mathbf{K}=(k_1,\ldots,k_L) \mid K = \sum_{1\le I\le L} k_I, \sum_{1\le I\le L} k_I \cdot w_I(T) \le H], \tag{6.1-3}$$

wenn sich K Aufträge unter Einhaltung der Randbedingung im Dispatching-System befinden.
Befindet sich ein Auftrag mit der Wahrscheinlichkeit π_I in der Phase I, dann ist die gesuchte Wahrscheinlichkeit
P_{k_1,k_L} **polynomial verteilt.** Ohne die Randbedingung zu beachten gilt:

$$\text{Prob}[\mathbf{K}=(k_1,\ldots,k_L)] = \frac{K!}{k_1!\cdot\ldots\cdot k_L!}\ \pi_1^{k_1}\cdot\ldots\cdot\pi_L^{k_L}. \tag{6.1-4}$$

Mit dieser Formel werden die Auftrittswahrscheinlichkeiten aller möglichen Zustandsvektoren berechnet. Insgesamt gibt es

$$|\mathbf{K}| = \binom{L+K-1}{K} = \binom{L+K-1}{L-1} \tag{6.1-5}$$

verschiedene Möglichkeiten, K Aufträge auf L Lastklassen zu verteilen. Allerdings ist bei (6.1-5) die Randbedingung (6.1-2) nicht berücksichtigt, durch die ja u.U. einige der möglichen $\mathbf{K}$-Vektoren ausgeschlossen werden. Um dies einzubeziehen, müssen die in (6.1-4) angegebenen Wahrscheinlichkeiten noch entsprechend normiert werden. Dazu definieren wir:

$$G := \sum_{\mathbf{K}} [P_{k_1,k_L} \mid \sum_{1\le I\le L} k_I \cdot \overline{w}_I(T) \le H]. \tag{6.1-6}$$

Damit erhalten wir schließlich für die gesuchte Wahrscheinlichkeit:

$$P_{k_1,k_L} = \frac{1}{G}\cdot\frac{K!}{k_1!\cdot\ldots\cdot k_L!}\ \pi_1^{k_1}\cdot\ldots\cdot\pi_L^{k_L}. \tag{6.1-7}$$

Um (6.1-7) anwenden zu können, müssen wir noch die stationären Wahrscheinlichkeit π_I bestimmen, mit der sich ein Auftrag in der Phase I befindet. D.h. das Phasenwechsel-Verhalten eines Auftrags muß modelliert werden.

Im Sinne der Taxonomie von Brandwajn /BRAN 85/ findet dazu eine Zustandsaggregierung statt, d.h. diejenigen Zustände eines Auftrags werden zusammengefaßt, bei denen er sich im Lokalitätsbereich I befindet (vgl. auch Abschnitt 3). Die Wahrscheinlichkeiten π_I, I=1,..,L, sind dann die **marginalen stationären Wahrscheinlichkeiten.**

Bestimmung der marginalen stationären Zustandswahrscheinlichkeiten π_I

Gegeben sei ein markovsches Modell des Seitenzugriffs mit der zugehörigen Transitionsmatrix $\mathbf{Q}$, die entsprechend Abschnitt 2.1 die NCD-Eigenschaft aufweist. Dann kann $\mathbf{Q}$ in eine Matrix $\mathbf{Q}^*$ mit Blockdiagonalform transformiert werden, wobei $\mathbf{Q}^*$ genau L quadratische Submatrizen $\mathbf{Q}_I^*$, mit I=1,..,L, enthält. Mit der Notation von Gleichung (2.1-9) sind nach /GEMI 80/ die Elemente der stochastischen quadratischen Submatrizen $\mathbf{Q}_I^*$ durch

$$q_{i(I)j(I)}^* = q_{i(I)j(I)} \Big/ \sum_{1 \le k \le m(I)} q_{i(I)k(I)} \qquad (6.1\text{-}8)$$

gegeben.
Durch Lösen des Gleichungssystems (2.1-11) lassen sich die stationären Zustandsvektoren $\boldsymbol{\pi}_I^* = (\pi_{1(I)}^*, .., \pi_{m(I)}^*)$ für $I \in \{1, \ldots, L\}$ bestimmen. Dabei ist $\pi_{i(I)}^*$ die approximierte stationäre Zugriffswahrscheinlichkeit auf die Seite s_i aus der Phase I.

Die Wahrscheinlichkeiten eines Phasenwechsels lassen sich leicht errechnen, wenn Matrix $\mathbf{Q}$ die **"lumpability"-Eigenschaft** besitzt, oder sie nach /COUR 77, S.37/ "*row block stochasticity*" aufweist, d.h. wenn gilt:

$$\sum_{j(J)} q_{i(I)j(J)} = \sum_{j'(J)} q_{i'(I)j'(J)} \quad \text{für alle } i(I),\ i'(I). \qquad (6.1\text{-}9)$$

Gleichung (6.1-9) ist dann erfüllt, wenn für alle Seiten aus Phase I die Wahrscheinlichkeit eines Wechsels zur Phase J gleich ist. In diesem Fall ist die Wahrscheinlichkeit p_{IJ} eines Phasenwechsel von I nach J durch:

$$p_{IJ} = \sum_{j(J)} q_{i(I)j(J)} \qquad (6.1\text{-}10)$$

gegeben. Die Seite i aus Phase I kann dabei wegen der Bedingung (6.1-9) beliebig gewählt sein.

Gilt jedoch die "lumpability"-Eigenschaft nicht, so kann nach /COUR 77/ die Wahrscheinlichkeit eines Phasenwechsels durch

$$p_{IJ} = \sum_{i(I)} \pi_{i(I)}^* \sum_{j(J)} q_{i(I)j(J)} \qquad (6.1\text{-}11)$$

abgeschätzt werden. Mit der so bestimmten Matrix

$$\mathbf{P} := [p_{IJ}] \qquad (6.1\text{-}12)$$

läßt sich durch Lösen des Gleichungssystems

$$\boldsymbol{\pi} = \mathbf{P} \cdot \boldsymbol{\pi} \qquad (6.1\text{-}13)$$

der stationäre Zustandsvektor $\pi=(\pi_1,..,\pi_L)$ bestimmen, wobei π_I die gesuchte stationäre Wahrscheinlichkeit ist, daß sich der Auftrag in Phase I, $1 \leq I \leq L$, befindet.

Mit (6.1-13) läßt sich also nach (6.1-7) abschließend die Zustandsverteilung des Lastvektors **K** bestimmen.

Die Lösung eines Systems mit sehr vielen Lastklassen ist allerdings problematisch, weil dann die Größe des Modell-Zustandsraums sehr stark ansteigt /BEIL 88/, /CHOW 83/. Zudem muß nach dem Verfahren der Dekompositions-Approximation für jeden möglichen Lastvektor ein eigenes Warteschlangen-Modell gelöst werden (vgl. Abschnitt 6.2).

Deshalb wird häufig nur vom einfachsten Lastmodell, nämlich nur **einer einzigen Lastklasse** ausgegangen.

Oft wird die Last auch dann durch nur eine einzige Klasse modelliert, wenn zu wenige Aussagen über die genaue Systemlast vorliegen, oder der Einfluß der Last nicht untersucht werden soll.

In diesem Fall, d.h. bei identischem Programmverhalten sämtlicher Aufträge, läßt sich die mittleren Anzahl K der speicherresidenten Aufträge leicht durch

$$\bar{K} = H \; / \; w(T) \qquad (6.1\text{-}14)$$

abschätzen.

6.2 Modell des Auftragsverhaltens

In diesem Abschnitt wird der Schritt der Dekomposition nach /BRAN 85/ durchgeführt: Unter der Bedingung, daß die Last im Dispatching-System durch einen bestimmten Lastvektor (vgl. Abschnitt 6.1) gegeben ist, werden die bedingten Zustands-Wahrscheinlichkeiten (conditional probabilities) des Systems bestimmt.

Modell-Voraussetzungen:

Dieser Modell-Teil geht davon aus, daß die Anzahl der im Dispatching-System vorhandenen, speicherresidenten Aufträge durch den Lastvektor

- $\mathbf{K} = (k_1, k_2, \ldots, k_L)$

vorgegeben ist. D.h. die Systemlast wird durch L verschiedene Lastklassen modelliert, wobei sich von jeder Lastklasse I genau k_I Aufträge im Dispatching-System befinden. Die Wahrscheinlichkeit, daß sich bei K Aufträgen diese Lastverteilung ergibt, ist durch (6.1-7) bestimmt.
Weiterhin setzen wir voraus, daß jeder in Phase I befindliche Auftrag durch

- die Bedienzeitanforderung B_I an die CPU und
- die Seitenzugriffs-Rate $ß_I$

spezifiziert ist.

In Anlehnung an die meisten Modelle zur Untersuchung der System-Leistung, wollen wir das Dispatching-System durch das folgende, in Abbildung 6.1 dargestellte Warteschlangen-Netzwerk (**central server model**) /BUZE 73/, /LAVE 83/ modellieren.

Die Station F(iling) modelliert den Datei-Transfer bei Ein-/ Ausgabe-Operationen, und die Station P(aging) das bei einem Seitenfehler erforderliche Seitenwechseln. Die Bedienzeit der Paging-Station entspricht der Zeit, die benötigt wird, eine Seite vom Plattenspeicher in den Hauptspeicher einzulagern und gegebenenfalls eine zu "opfernde" Seite auszulagern.

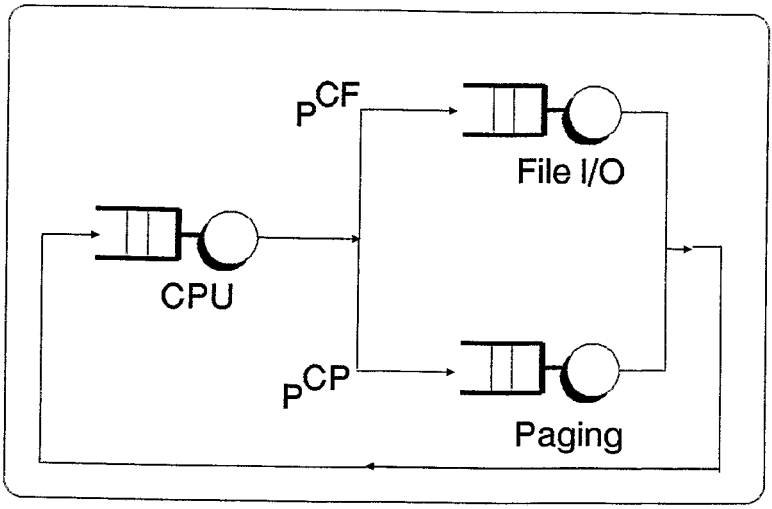


Abb. 6.1: Warteschlangen-Netzwerk zur Modellierung des Leistungs-Verhaltens

Nach Voraussetzung besitzt jeder Auftrag der Phase I eine mittlere Bedienanforderung von B_I. In Systemen mit Mehrprogrammbetrieb (multiprogramming) steht aber die CPU einem Auftrag nicht ununterbrochen für diesen Zeitraum zur Verfügung. Vielmehr wird ihm durch bestimmte Ereignisse, wie z.B. Seitenfehler, die CPU entzogen.

Deshalb kann sich jeder Auftrag in einem der drei folgenden Zustände befinden:

- **running**: Er wird von der CPU bearbeitet,
- **ready**: Er wartet in der CPU-Warteschlange auf die Prozessor-Zuteilung,
- **suspended**: Für ihn muß noch ein Seitenwechsel oder ein Datei-Transfer an einer der Stationen P bzw. F durchgeführt werden.

Die Zustandsübergänge geschehen also nach dem Zustandsgraphen von Abbildung 6.2.

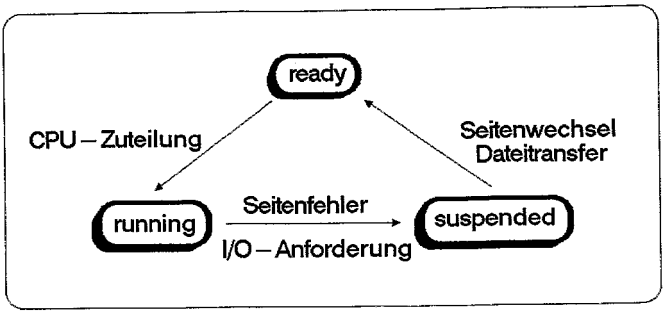


Abb. 6.2: Wechsel der Auftragszustände im Dispatching-System

Mit dem in Abbildung 6.1 dargestellten Warteschlangen-Netz haben wir das Dispatching-System auf drei Stationen reduziert. Die verbleibende Aufgabe ist es nun, die Modell-Parameter unter Berücksichtigung von Speicher- und Fehlerverhalten zu spezifizieren. Dies muß für jede **Lastklasse** I, die das Verhalten eines Auftrags in Phase I modelliert, gesondert geschehen.

A. Speicherverhalten: Bestimmen der Paging-Rate und der Arbeitsmengengröße

Zugrunde gelegt wird die Spezifikation des Programmverhaltens für die Phase I mit Hilfe eines IRM bzw. MRM-Modells (vgl. Abschnitte 2.1 und 6.1). Dann läßt sich für die Submatrix Q_I von Phase I der stationäre Zustandsvektor $\pi(I)^*=(\pi_{1(I)}^*,..,\pi_{m(I)}^*)$ berechnen. Die durch $\pi_{i(I)}^*$ gegebenen stationären Zugriffswahrscheinlichkeiten jeder einzelnen Seite $s_{i(I)}$ einer Phase I können gemäß (2.1-3) ein IRM-Modell spezifizieren, mit

$$b_i := \pi_i^* \ , \ i=1,...,m \ . \qquad (6.2\text{-}1)$$

In (6.2-1) verzichten wir auf den Phasen-Index I, um die Notation zu vereinfachen.

Zur Bestimmung der Paging-Rate können wir von Gleichung (5.2-4) ausgehen. Summiert man dort über alle Seiten der Phase I, so erhält man:

$$\Phi^{P}_{I} = \sum_{1 \le i \le m(I)} \bar{C}_{i,1}/B_I$$

$$= \sum_{1 \le i \le m(I)} [\pi_i(1-\pi_i)^T \cdot \beta_I + (1-(1-\pi_i)^T)/B_I] \quad . \qquad (6.2\text{-}2)$$

Dabei ist zu beachten, daß in (6.2-2) durch Fehler induzierte zusätzliche Seiteneinlagerungen noch nicht berücksichtigt sind.
Die durchschnittliche **Arbeitsmengen-Größe** läßt sich nach /DESC 72/ für das IRM-Modell mit geringem Aufwand durch:

$$\bar{w}_I(T) = m(I) - \sum_{1 \le i \le m(I)} (1 - \pi_{i(I)})^T \qquad (6.2\text{-}3)$$

berechnen. In /DESC 72/ wird zusätzlich eine (hier nicht benötigte) Formel zur Abschätzung der Varianz genannt. Wichtiger ist für uns die Wahrscheinlichkeits-Verteilung der Arbeitsmengengröße. Definieren wir mit

$$g(k,T,m) := \text{Prob}[w(T,m)=k] \quad , \qquad (6.2\text{-}4)$$

die stationäre Wahrscheinlichkeit, mit der die Arbeitsmenge eine Größe von k Seiten aufweist, dann errechnet sich die Verteilung von k nach /VANT 74/ durch:

$$g(k,T,m) = \sum_{0 \le h \le k} (-1)^{k-h} \binom{m-h}{m-k} D(h,T,m(I)) \quad ,$$

wobei

$$D(h,T,m) = \sum_{h\text{-Tupel}} (\pi_{i(1)}+\ldots+\pi_{i(h)})^T \quad , \qquad (6.2\text{-}5)$$

mit $i(k) < i(k+1)$.

In der oben genannten Arbeit wird ebenfalls nachgewiesen, daß die Arbeitsmengengröße approximativ **normalverteilt** ist.

B. Belegungsdauer der CPU

Die mittlere Zeitdauer $(\mu^{CPU})^{-1}$ einer **CPU-Belegung ohne Unterbrechung** ist eine der kritischen Größen des Modells mit starken Auswirkungen auf das Leistungsverhalten.
Sobald ein Auftrag der CPU zugeteilt ist, können vier Ereignisse eintreten, die jeweils zu einer Unterbrechung, also einem Entzug der CPU führen, sodaß der Auftrag in den Zustand **suspended** wechselt.

Diese vier Ereignisse sind:

- Anforderung einer I/O-Operation (mit der Rate $\Phi^{I/O}$),
- Auftreten eines regulären Seitenwechsels (mit der Rate Φ^{P}),
- Seitenwechsel aufgrund eines Paging- oder Hauptspeicherfehlers (mit der Rate $\Phi^{PF}+\Phi^{HF}$),
- Ablaufen einer CPU-Zeitscheibe (time slice).

Die Überlagerung der den oberen drei Ereignissen zugeordneten stochastischen Prozesse bildet wiederum einen stochastischen Punkt-Prozeß. Um diesen resultierenden Prozeß im Rahmen unseres Modells (einfach) bestimmen zu können, sei im folgenden vorausgesetzt, daß die genannten Prozesse Poisson-Verteilungen besitzen. Dann ist der überlagerte Prozeß wiederum ein Poisson-Prozeß /LAVE 83/ mit der mittleren Rate μ^{CPU}, für die gilt:

$$\mu^{CPU} = \Phi^{I/O} + \Phi^{P} + \Phi^{HF} + \Phi^{PF} \quad . \qquad (6.2\text{-}6)$$

Die Rate der I/O-Operationen $\Phi^{I/O}$ ist eine Programm-Charakteristik des Auftrags und ist somit von den betrachteten Fehler-Arten und von der aktuellen Arbeitslast unabhängig. Die Paging-Rate Φ^{P} ist durch Gleichung (6.2-2) gegeben, wobei durch Fehler verursachte zusätzliche Seiteneinlagerungen noch unberücksichtigt sind. Deren Einbeziehung erfolgt durch die in Kapitel 5 abgeleiteten Raten Φ^{HF} und Φ^{PF}.
Mit Hilfe von CPU-Zeitscheiben wird das in Dispatching-Systemen allgemein übliche Round-Robin-Verfahren zur Prozessor-Zuteilung realisiert. Es wird im Grenzfall sehr kleiner Zeitscheiben bei Central-Server-Modellen durch die Bedienstrategie PS (**processor sharing**) modelliert /LAVE 83/. Das ist vor allen Dingen in (Dialog-)Systemen realistisch, wenn die Rate, mit der Zeitscheiben ablaufen, um ein Vielfaches kleiner als die Seitenfehler-Rate ist. Um eine gute Lösbarkeit des Modells zu gewährleisten, sollen auch in unserem Modell an der CPU-Station Aufträge mit der Bediendisziplin processor sharing bedient werden.

C. Einbinden von Hauptspeicherfehlern ins WS-Modell

Das oben beschriebenen Warteschlangen-Modell dient dazu, den Auftragsfluß im System abzubilden und entsprechende Leistungsindizes zu berechnen. Bei dieser Modell-Sicht lassen sich Fehler vollständig anhand ihrer Auswirkungen auf das Leistungsverhalten darstellen. Diese Auswirkungen können durch die zur Fehlerbehandlung erforderliche zusätzliche Last charakterisiert werden. Dabei sind mehrere

grundlegende Effekte zu berücksichtigen (vgl. Abbildung 2.4, sowie die Abschnitte 5.2-5.4).

- Zur Wiederherstellung der Datenintegrität müssen die betroffenen und zusätzlich eventuell verfälschte Seiten erneut eingelagert werden, d.h. die **Gesamt-Paging-Rate** wird durch das Auftreten von Fehlern **erhöht**. (Die zugehörigen Raten wurden für verschiedene Verfahren der Fehlerentdeckung in Kapitel 5 berechnet.)

- Aufgrund der höheren Paging-Rate verringert sich die **mittlere Zeit einer CPU-Belegung** ohne Unterbrechung $(\mu^{CPU})^{-1}$ (vgl. (6.2-6)).

- Ein Auftrag muß gegebenenfalls ab einem fehlerfreien Zustand (teilweise) wiederholt werden. Dadurch **erhöht** sich die **Bedienanforderung** B_I.

Es ist zu beachten, daß Parameter, wie die Paging-Rate und die Verweildauern keine Charakteristika des einzelnen Auftrags sind, sondern von der Systemlast und den gewählten Betriebssystem-Parametern (bspw. von der Fenstergröße) abhängen. Sie müssen folglich im Leistungsmodell bestimmt werden.

Insgesamt werden durch Fehler offensichtlich die Verweildauern und die Antwortzeiten der betroffenen Aufträge erhöht, der Durchsatz sinkt also entsprechend. Dabei bestehen die schon in Abbildung 2.4 dargestellten Wechselwirkungen: Die Speicherfehler-Rate Φ^F wächst mit der Verweildauer V_I eines Auftrags in Phase I. Diese wird jedoch, wie oben festgestellt, durch jeden Fehler erhöht. So verlängert sie sich u.a. um die Wartezeit auf einen Seitenwechsel und die für den eigentlichen Einlagerungsvorgang benötigte Zeit.
Ein ähnlicher Zusammenhang betrifft die Paging-Fehler. Je mehr Seitenwechsel erfolgen, um so mehr Paging-Fehler treten auf, die wiederum mehr Seitenwechsel verursachen.

Diese **rekursiven Abhängigkeiten** müssen bei der Einbindung des Fehlermodells in das Warteschlangenmodell berücksichtigt werden. Das dazu erforderliche rekursive Berechnungs-Schema wird im nächstem Kapitel vorgestellt.

D. Spezifikation der Übergangswahrscheinlichkeiten

In unserem Warteschlangen-Modell müssen nun noch die Übergangswahrscheinlichkeiten bestimmt werden. Wegen des zugrunde gelegten Central-Server-Modells sind dabei ausschließlich die Übergänge beim Verlassen der CPU zu betrachten.

Für jede Lastklasse I errechen sich die Wahrscheinlichkeiten p^{CF} und p^{CP} aus den Übergangsraten zur File- und Paging-Station sowie der Speicherfehler-Rate. Es gelten:

$$p^{CF} = \frac{\Phi^{I/O}}{\Phi^{I/O} + \Phi^{P} + \Phi^{HF} + \Phi^{PF}} = \frac{\Phi^{I/O}}{\mu^{CPU}} \qquad (6.2\text{-}7)$$

und

$$p^{CP} = 1 - p^{CF} = \frac{\Phi^{P} + \Phi^{HF} + \Phi^{PF}}{\Phi^{I/O} + \Phi^{P} + \Phi^{HF} + \Phi^{PF}} \quad . \qquad (6.2\text{-}8)$$

Im Nenner beider Gleichungen stehen die Raten sämtlicher Ereignisse, die zu einem Prozessor-Entzug führen können. In (6.2-8) stehen im Zähler auch die Raten Φ^{HF} und Φ^{PF} der durch Fehler verursachten Seiteneinlagerungen, weil diese ebenfalls zu einer Anforderung an die Paging-Station führen.

E. Spezifikation des I/O-Verhaltens

Die **Bedienstrategie** an der File- und Paging-Station soll jeweils FCFS sein und es soll mit den mittleren Bedienraten μ^{P} und $\mu^{I/O}$ bedient werden. Nach den Voraussetzungen des BCMP-Theorems /BCMP 75/ können dann nur exponential verteilte Bedienzeiten zugelassen sein.

Statt einer lastunabhängigen Bedienzeitverteilung könnte auch eine lastabhängige Bedienzeitverteilung gewählt werden, z.B. das sogenannte Eschenbach drum scheme /WEIN 68/. Dabei wird angenommen, daß Seitenwechsel-Anforderungen nicht in der Reihenfolge ihrer Ankunft bedient werden, sondern daß zunächst diejenige Seite eingelagert wird, die sich in dem zum Lese-/Schreibkopf nächstgelegenen Sektor befindet. Nach /WEIN 68/ ergibt sich für die Bedienrate $\mu^{P}(n_p)$, die von der Anzahl n_p der auf Seitenwechsel wartenden Aufträge abhängt:

$$\mu^{P}(n_p) = \frac{2 \cdot n_p}{S + 2 \cdot n_p - 1} \cdot \frac{S}{T_{rot}} \quad , \qquad (6.2\text{-}9)$$

mit der Anzahl S der Sektoren und der Rotationszeit T_{rot} des Speichermediums .

F. Test-Overhead

In unserem Modell bleibt der Overhead für einen Speichertest bisher unberücksichtigt. Insbesondere wenn dieser Test durch Software realisiert ist, führt jedoch jede Testdurchführung u.a. zu einer Verlängerung der Bedien- und der Verweildauer.

Werden C_I Speichertests in Phase I durchgeführt, und erfordert jeder Test eine zusätzliche Bedienzeitdauer B^T, dann erechnet sich die effektive Bedienzeitanforderung B^{eff} durch:

$$B^{eff} = B_I + C_I \cdot B^T \quad . \tag{6.2-10}$$

Selbstverständlich auch Auswirkungen des Test-Overheads auf andere Größen, z.B. auf die Paging-Rate modelliert werden.

7 Zur Integration der Teilmodelle

In diesem Kapitel sollen die in den drei vorhergehenden Kapiteln entwickelten Teilmodelle in ein Gesamtmodell integriert werden. Dabei werden zunächst mit einem iterativen Lösungsverfahren die Wechselwirkungen zwischen den einzelnen Teilmodellen berücksichtigt. Im Anschluß daran werden einige Überlegung zur Terminierung der Iteration diskutiert. Der letzte Abschnitt dieses Kapitels versucht, Aussagen über das Auftreten und die Wirkung von Fehlern bei einem einzelnen Auftrag zu machen.

7.1 Iteratives Lösungsverfahren

In diesem Abschnitt wollen wir die wichtigsten Ergebnisse aus den Kapiteln 5 und 6 miteinander verbinden.
Nach der Modellentwicklung lassen sich offensichtlich die in Abbildung 7.1 dargestellten vier Modellierungs-Phasen A, B, C und D unterscheiden.

A. Spezifikation des Gesamt-Modells
- A.1 Auftragsfluß
- A.2 Speicherverwaltung
- A.3 Störungen

B. Lastmodell
- B.1 Anzahl der Phasen
- B.2 Phasenwechsel-Verhalten
- B.3 Lastverteilung

iterativ:

C. Leistungsmodell
- C.1 Bedienanforderung
- C.2 Bedienraten
- C.3 Paging-Rate
- C.4 Übergangswahrscheinlichkeiten
- C.5 Lösung des Warteschlangen-Netzes

D. Fehlermodell
- D.1 Auftreten von Störungen
- D.2 Fehlerbehandlung

Abb. 7.1: Phasen der Modellierung

Die ersten beiden **Teilschritte A und B** spezifizieren die Lastcharakteristika der Aufträge und sind keinen Wechselwirkungen mit dem Fehlermodell ausgesetzt. Deshalb müssen sie **nicht iteriert** werden.
Wie in den Kapiteln 2 und 3 dargelegt, bestehen jedoch Wechselwirkungen zwischen den Teilmodellen C und D.

Leistungsmodell
Ins Leistungsmodell gehen für jede Phase (Lastklasse) I die folgenden Parameter ein, die erst im Fehlermodell bestimmt werden:

- B^{R}_{I}: die durch Fehler verursachte Verlängerung der Bedienzeitanforderung,
- Φ^{F}_{I}: die Auftrittsrate von Hauptspeicherfehlern,
- Φ^{HF}_{I}: die durch Hauptspeicherfehler induzierte zusätzliche Paging-Rate,
- Φ^{PF}_{I}: die durch Paging-Fehler induzierte zusätzliche Paging-Rate.

Diese Größen quantifizieren die durch Fehler induzierte zusätzliche Last.

Fehlermodell
Ins Fehlermodell aus den Kapiteln 4 und 5 gehen die im Leistungsmodell bestimmte

- Verweildauer V_I, sowie
- die Paging-Rate Φ^{P}_{I}

ein.

Der rekursive Zusammenhang zwischen den einzelnen Teilmodellen wurde im Kapitel 3 formal durch die Gleichungen (3.2-1) bis (3.2-3) dargestellt. Die Modell-Lösung wird durch die Fixpunkt-Gleichung (3.2-4) definiert.

Um Fixpunkte linearer Gleichungen bestimmen zu können, werden normalerweise **iterative Verfahren** /ORRH 70/ angewandt.
Ähnlich wie beim MVA-Lösungsverfahren (**mean value analysis**) /REIS 79/, /RELA 80/ werden dabei in jedem Iterations-Schritt Parameter errechnet, die dann als Eingaben des nächsten Iterations-Schritts dienen. Die große Bedeutung iterativer Lösungsmethoden für

die Leistungsbewertung von Rechensystemen wird in /SSLM 84/ dargestellt.

Durchführung der Iteration

Beim **ersten Iterations-Schritt** wird das Leistungsmodell ohne Berücksichtigung von Fehlern behandelt. Dabei werden die oben genannten, im Fehlermodell zu bestimmenden Parameter des Leistungsmodells jeweils mit 0 initialisiert.
Bei jedem Iterations-Schritt wird das Leistungsmodell C mit den Parametern des vorhergehenden Iterations-Schrittes spezifiziert. Als Ergebnis des Leistungsmodells erhält man insbesondere die Verweildauer und die Paging-Rate jeder Phase. Mit diesen Größen kann dann das Fehlermodell spezifiziert werden. Die Raten der zusätzlichen Seiteneinlagerungen und die Verlängerung der Bedienzeit liefern neue (Last)-Parameter für das Leistungsmodell, die im nächsten Iteration-Schritt verwendet werden. Das Verfahren stoppt, sobald die Differenz zwischen den in sukzessiven Iterations-Schritten berechneten Maßen genügend klein ist:

$$|V^{(n)} - V^{(n-1)}| < \varepsilon \ .$$

Die eingeklammerte Hochzahl $^{(n)}$ verweist dabei im folgenden auf eine im n-ten Iterations-Schritt berechnete Größe. Zur Vereinfachung der Notation werden die Mittelwert-Striche im folgenden weggelassen. Der Algorithmus wird in Abbildung 7.2 schematisch dargestellt.

Zu beachten ist, daß für jede durch einen Vektor $\mathbf{K} = (k_1,\ldots,k_R)$ gegebene Lastkonfiguration eine eigene Iteration durchgeführt werden muß.

Initialisierung:

$B^{R_I(0)} := 0$, $\Phi^{F_I(0)} := 0$, $\Phi^{HF_I(0)} := 0$, $\Phi^{PF_I(0)} := 0$, n:=1

REPEAT

Iterationsschritt n:

C. Spezifiziere das **Leistungsmodell**. Benutze dabei als Eingabegrößen $B^{R_I(n-1)}$, $\Phi^{F_I(n-1)}$ und $\Phi^{PF_I(n-1)}$ des vorhergehenden Iterationsschritts (n-1).
Löse das Warteschlangen-Netz für jede mögliche Konfiguration **K** und bestimme die **Verweildauer** $V_I^{(n)}$ und die Paging-Rate Φ^{P}_I eines Auftrags für jede Phase I.

D. Löse mit der neu berechneten Verweildauer $V_I^{(n)}$ die Gleichungen des Fehlermodells. Bestimme dabei die Größen $B^{R_I(n)}$, $\Phi^{F_I(n)}$, $\Phi^{HF_I(n)}$, und $\Phi^{PF_I(n)}$.

n:=n+1;

UNTIL $|V_I^{(n)} - V_I^{(n-1)}| > \varepsilon$

Abb. 7.2: Konzeptionelles Vorgehen bei der Iteration

Im folgenden wollen wir noch einmal genauer auf die einzelnen Iterationsschritte eingehen. Dies zeigt wie die Teilmodelle ineinander greifen, und ist auch gleichzeitig eine Zusammenfassung unseres Modellbildes aus den vorhergehenden Kapiteln.
Zuerst werden die beiden nicht zu iterierenden Teilschritte A und B durchgeführt.

A. Spezifikation des Gesamt-Modells

In diesem Teil werden noch einmal die Voraussetzungen unseres Modells zusammengefaßt.

A.1 Auftragsfluß

Der Auftragsfluß in unserem Modell wird durch die folgenden Voraussetzungen spezifiziert.

A.1.1 Untersucht wird ein Dispatching-System, das aus den drei Bedienstationen CPU, FILE-I/O und PAGING besteht (siehe Abbildung 6.1).

A.1.2 Jeder im Dispatching-System befindliche Auftrag stellt an die CPU eine Bedienzeitanforderung von B^G Zeiteinheiten.

A.1.3 Das von der Systemlast unabhängige Ein-/ Ausgabeverhalten ist durch die Seitenzugriffs-Rate ß und die I/O-Rate $\Phi^{I/O}$ gegeben.

A.1.4 Die Bedienraten an der I/O- und der Paging-Station sind durch $\mu^{I/O}$ bzw. μ^{P} vorgegeben.

A.2 Speicherverwaltung

A.2.1 Die Speicherverwaltung arbeitet nach dem working-set-Prinzip mit der gegebenen Fenstergröße von T Seiten. Die Speichergröße beträgt H Seiten.

A.2.2 Der Zugriff auf einzelne Speicherseiten eines Auftrags soll mit Hilfe eines markovschen Referenz-Modells (MRM) durch eine Übergangs-Matrix **Q** wie in (2.1-6) beschrieben werden.

A.3 Störungen

Das Auftreten von Störungen in der Speicherverwaltung wird durch die in Kapitel 4 genannten Parameter festgelegt.

A.3.1 **Ereignisunabhängige Störungen:**
Der zeitliche Abstand s zweier Störungen des Haupt-speichers wird durch die Verteilungsfunktion $F_S(t)$, siehe (4.1-1), spezifiziert.
Die Hardware-Architektur des Speichers bestimmt nach Abschnitt 4.1.2 die Verteilung e(k) der Anzahl k der von einer bestimmten Störung betroffenen Seiten.

A.3.2 **Ereignisabhängige Störungen:**
Bei zugriffsinduzierten Störungen ist p_z die Wahrscheinlichkeit, mit der ein Seitenzugriff gestört wird.
Die Wahrscheinlichkeit von Seitenwechsel-induzierten Störungen ist p_p.

B. Lastmodell

In diesem Teil des Algorithmus wird die System-Last bestimmt. Hierbei interessiert uns insbesondere das Speicherverhalten der Aufträge. Es werden die folgenden drei Teil-Schritte durchgeführt.

B.1 Anzahl der Phasen

Zunächst muß bestimmt werden, wieviele verschiedene Phasen (Lokalitätsbereiche) ein einzelner Auftrag aufweist.

Dazu wird die Übergangs-Matrix $\mathbf{Q}$ in eine Matrix mit Blockdiagonalform $\mathbf{Q}^*$ gemäß (2.1-7) und (2.1-8) transformiert. Die Anzahl L der quadratischen Submatrizen entspricht dann der Phasen-Anzahl eines Auftrags.

B.2 Phasen-Verhalten

In diesem Schritt wird die marginale stationäre Wahrscheinlichkeit π_I berechnet, mit der sich ein Auftrag in Phase I befindet.

B.2.1 Dazu wird als erstes mit Hilfe von (6.1-10) oder (6.1-11) die Phasen-Übergangsmatrix Matrix $\mathbf{P}$ generiert.

B.2.2 Anschließend wird die Lösung des Gleichungssystems (6.1-13) $\boldsymbol{\pi} = \mathbf{P} \cdot \boldsymbol{\pi}$, d.h. der stationäre Zustandsvektor $\boldsymbol{\pi}=(\pi_1,..,\pi_L)$ bestimmt.

B.2.3 Wie in (2.1-11) angegeben, kann schließlich durch Lösen des Gleichungssystems $\boldsymbol{\pi}_I^* = \mathbf{Q}_I^* \cdot \boldsymbol{\pi}_I^*$ (wobei gilt $(\boldsymbol{\pi}_I^*=(\pi_{1(I)}^*,..,\pi_{m(I)}^*)$ die stationäre Zugriffswahrscheinlichkeit $\pi_{i(I)}$ einer einzelnen Seite s_i in Phase I bestimmt werden. Die Wahrscheinlichkeiten $\pi_{i(I)}^*$ spezifizieren für jede einzelne Phase ein independent reference model (IRM), siehe Abschnitt 2.1.

B.3 Lastverteilung

In diesem Schritt wird die Wahrscheinlichkeit berechnet, mit der sich im Dispatching-System eine bestimmte Last befindet.

B.3.1 Zunächst werden für jede Phase I die mittleren Arbeitsmengengrößen berechnet. Mit (6.2-5) ergibt sich:

$$\overline{w}_I(T) = m(I) - \sum_{1 \le i \le m(I)} (1-\pi_i(I))^T \ .$$

B.3.2 Unter der Annahme, daß sich insgesamt K Aufträge im Dispatching-System befinden, werden alle Lastvektoren $\mathbf{K} = (k_1,..,k_L)$ bestimmt, welche die Randbedingung (6.1-2) erfüllen, d.h. für die gilt:

$$\sum_{1 \le I \le L} k_I \cdot \overline{w}_I(T) \le H \ .$$

B.3.3 Mit (6.1-4) bis (6.1-7) erhält man für jeden in B.3.2 bestimmten Lastvektor die Wahrscheinlichkeit seines Auftretens als

$$P_{k_1,k_L} = \frac{1}{G} \cdot \frac{K!}{k_1! \cdot k_2! \cdot \ldots \cdot k_L!} \pi_1^{k_1} \cdot \ldots \cdot \pi_L^{k_L} .$$

Um die Wechselwirkungen zwischen Leistungs- und Fehlergrößen zu berücksichtigen, werden die folgenden Teilschritte C und D **iterativ** ausgeführt.

C. Leistungsmodell

Für jeden in B.3.2 bestimmten zulässigen Lastvektor $\mathbf{K} = (k_1,..,k_L)$ werden durch ein eigenes Warteschlangen-Netz Leistungsindizes bestimmt.
Diese Lastkonfiguration wird mit der Wahrscheinlichkeit P_{k_1,k_L} angenommen.
Für jede Auftragsphase wird eine **Lastklasse** I=1,..,L, im Warteschlangen-Netz wie in den Punkten C.1-C.5 spezifiziert. (Die in Klammern hochgestellte Zahl n verweist wieder auf eine Größe, die im n-ten Iterations-Schritt berechnet wurde.)

Es werden die folgenden **Initialisierungen** benutzt:
$B^{R_I(0)} := 0$, $\Phi^{F_I(0)} := 0$, $\Phi^{HF_I(0)} := 0$, $\Phi^{PF_I(0)} := 0$.

C.1 Bedienzeitanforderung

Ein Last-Charakteristikum jedes Auftrag ist seine Bedienzeitanforderung B^G an die CPU. Wenn jeder Auftrag sich mit der Wahrscheinlichkeit π_I in Phase I befindet, beträgt im ungestörten Fall für die Bedienzeitanforderung B_I in Phase I:

$$B_I = \pi_I \cdot B^G .$$

Mit dem vorhergehenden Iterations-Schritt ergibt sich für die durch Speicherfehler verursachte Verlängerung der Bedienzeitanforderung $B^{R_I(n)}$:

$$B_I^{(n)} = \pi_I \cdot B^G + B^{R_I(n-1)} \quad . \tag{7.1-1}$$

In Gleichung (7.1-1) lassen sich die Auswirkungen des Fehlermodells auf das Leistungsmodell erkennen.

C.2 Paging-Rate

Die Rate, mit der reguläre Seitenwechsel durchgeführt werden, ergibt sich aus der Gleichung (6.2-2), welche die Paging-Rate Φ^{P_I} ohne Speicherfehler berechnet.

$$\Phi^{P_I(n)} = \sum_{1 \le i \le m(I)} [\pi_i (1-\pi_i)^T \cdot \beta_I + (1-(1-\pi_i)^T)/B_I] \quad . \tag{7.1-2}$$

C.3 Bedienraten

Für sämtliche drei Stationen (vgl. Abb. 6.1) müssen die Bedienraten festgelegt werden. Nach Gleichung (6.2-8) ergibt sich für die Bedienrate der CPU:

$$\mu^{CPU_I(n)} = \Phi^{I/O_I} + \Phi^{P_I} + \Phi^{HF_I(n-1)} + \Phi^{PF_I(n-1)} \tag{7.1-3}$$

Die Raten Φ^{I/O_I} und Φ^{P_I} in (7.1-3) sind von Fehlern unabhängig und ändern sich deshalb nicht mit den Iterations-Schritten.
Wie in A.1.4 vorausgesetzt, sind die Bedienraten für die I/O- und die Paging-Station $\mu^{I/O}$, bzw. μ^P vorgegeben.

C.4 Übergangswahrscheinlichkeiten

Die in unserem Central-Server-Modell relevanten Übergangswahrscheinlichkeiten sind für jede Phase I durch (6.2-7) und (6.2-8) gegeben als

$$p^{CF_I(n)} = \frac{\Phi^{I/O}}{\mu^{CPU_I(n)}} \quad \text{und} \quad p^{CP_I(n)} = 1 - p^{CF_I(n)}. \tag{7.1-4}$$

C.5 Berechnung von Leistungsgrößen

Nachdem das Warteschlangen-Netz durch die Schritte C.1 bis C.4 für jede Lastklasse spezifiziert worden ist, müssen für jeden Lastvektor $\mathbf{K}$ mit Hilfe eines adäquaten Lösungsverfahrens Leistungsindizes bestimmt werden /BOAK 82/, /LAVE 83/. Die Berechnungen dieser Arbeit

werden mit der mean value analysis (MVA) /REIS 79/, /LARE 80/ durchgeführt.
Als Ergebnis eines bestimmten Warteschlangen-Netzes erhält man u.a. für jede Lastklasse I die mittlere Verweildauer V_I eines Auftrags, die von der Last $\mathbf{K} = (k_1,..,k_L)$ im Dispatching System abhängt.

D. Fehlermodell

Um im Leistungs-Modell die Gleichungen (7.1-1) bis (7.1-4) zu spezifizieren, werden Ergebnisse des Fehlermodells benötigt.

Für jede Phase I, mit I=1,..,L , muß ein eigenes Fehlermodell gelöst werden. Wir betrachten das Auftreten und die Auswirkung von Speicherfehlern. Dabei wird eines der in Abschnitt 5.1 vorgestellten Verfahren der Fehlerentdeckung vorausgesetzt.

Grundlage des Fehlermodells und **Schnittstelle zum Leistungsmodell** sind u.a. die im selben Iterations-Schritt bestimmten **Verweildauern** $V_I^{(n)}$ eines Auftrags. Im Fehlermodell werden sie insbesondere zur Berechnung der Zeitdauer D_i zwischen zwei Fehlerüberprüfungen benötigt.

D.1 Auftreten von Störungen

In unserem Störungsmodell aus Kapitel 4 wird zwischen ereignisunabhänigigen und zugriffsinduzierten Störungen unterschieden.

D.1.1 Als erstes wird für jede Seite s_i aus Phase I die Zeitdauer zwischen zwei aufeinanderfolgenden Fehlerüberprüfungen $D_i^{(n)}$ bestimmt.
Für eine Fehlerentdeckung

- beim **Seitenauslagern** ist dies die Einlagerungsdauer D^{E_i} (siehe (5.2-5)):

$$D_i^{(n)} := D^{E_i(n)} = \frac{[1 - (1-\pi_i)^T] \cdot V_I^{(n)}}{\pi_i \cdot (1-\pi_i)^T \cdot B_I^{(n)} \beta_I + [1-(1-\pi_i)^T]} \qquad (7.1\text{-}5a)$$

- beim **Zugriff** ist dies die Inter-Referenz-Zeit D^{R_i} (siehe (5.3-1)):

$$D_i^{(n)} := D^{R_i(n)} = \frac{[1 - (1-\pi_i)^T] \cdot V_I^{(n)}}{\pi_i \cdot \beta_i \cdot B_I^{(n)}} \qquad (7.1\text{-}5b)$$

durch **Konsistenzüberprüfung** in den Zeitabständen B^K ist dies die (seitenunabhängige) Inter-Konsistenzcheck-Zeit D^K (5.4-2):

$$D^{(n)} := D^{K(n)} = \frac{B^K}{B_I^{(n)}} \cdot V_I^{(n)} \quad . \qquad (7.1\text{-}5c)$$

D.1.2 Anschließend kann für **ereignisunabhängige Störungen** die Wahrscheinlichkeit $q^{U_i(n)}$ berechnet werden, daß mindestens eine Störung während $D_i^{(n)}$ auftritt. Sie errechnet sich, vgl. (5.2-10), (5.3-4) und (5.4-4), durch:

$$q^{U_i(n)} = \sum_{1 \le k \le \infty} Pr^U(k, D_i^{(n)}) \sum_{1 \le j \le k} (-1)^j \binom{j}{k} \cdot p_s^j \quad .$$

$Pr^U(k, D_i^{(n)})$ ist die Wahrscheinlichkeit, daß es k Störungspunkte in $D_i^{(n)}$ gibt, und wird in (4.1-9) errechnet. Die Wahrscheinlichkeit, mit der eine überprüfte Seite von einer einzelnen Störung betroffenen ist, wird in (5.2-9) bzw. (5.4-5) bestimmt.

D.1.3 Bei **zugriffsinduzierten Störungen** wird zunächst die mittlere Anzahl $Z^{C_i(n)}$ der Zugriffe berechnet, die während des Zeitraums $D^{(n)}$ auf Seite s_i stattgefunden haben. Sie sind für die verschiedenen Verfahren in (5.2-13) und (5.4-7) berechnet worden. Werden Fehler bereits beim Zugriff entdeckt (Abschnitt 5.3), dann gilt offensichtlich $Z^{C_i}=1$. Die Wahrscheinlichkeit $q^{Z_i(n)}$, daß bei einer Überprüfung der Seite ein zugriffsinduzierter Fehler entdeckt wird, ist:

$$q^{Z_i(n)} = 1 - (1 - p_z)^{Z^{C_i(n)}} ,$$

(vgl. (5.2-16), (5.3-7), (5.4-9)).

D.1.4 Da eine Seite s_i von einer der beiden Störungen mit der Wahrscheinlichkeit $q_i^{(n)}$ gemäß (5.2-17) betroffen wird, gilt nach der Summenregel der Wahrscheinlichkeits-Rechnung :

$$q_i^{(n)} = q^{U_i(n)} + q^{Z_i(n)} - q^{U_i(n)} \cdot q^{Z_i(n)} \quad .$$

D.1.5 Die Fehlerrate $\Phi^{F_i(n)}$ einer einzelnen Seite erhält man, indem die mittlere Anzahl $C_i^{(n)}$ der Fehlerüberprüfungen von Seite s_i (siehe (5.2-4), (5.3-9) bzw. (5.4-10)) mit der Wahrscheinlichkeit $q_i^{(n)}$ wichtet und durch $B_I^{(n)}$ dividiert wird, d.h.

$$\Phi^{F_i(n)} = q_i^{(n)} \cdot C_i^{(n)} / B_I^{(n)} .$$

Die Gesamtfehler-Rate $\Phi^{F(n)}$ der Phase I ist dann die Summe aller Einzelfehler-Raten:

$$\Phi^{F_I(n)} = \sum_{1 \le i \le m(I)} \Phi^{F_i(n)} = \sum_{1 \le i \le m(I)} q_i^{(n)} \cdot C_i^{(n)} / B_I^{(n)} \qquad (7.1\text{-}6)$$

D.2 Fehler-induzierte Last

Entdeckte Fehler werden durch (teilweises) Wiederholen des Auftrags sowie durch zusätzliche Seiteneinlagerungen behandelt.

D.2.1 Wird bei einer Überprüfung der Seite s_i ein Fehler entdeckt, dann muß der Auftrag um die Bediendauer $B^{R_i(n)}$ wiederholt werden:

$$B^{R_i(n)} = B_I^{(n)} \cdot D_i^{(n)} / V_I^{(n)} \approx a_i \cdot B_I^{(n)} / C_i^{(n)},$$

(vgl. (5.2-21), (5.4-15)).
Werden Fehler beim Zugriff entdeckt, so ergibt sich der Sonderfall $B^{R_i(n)} = 0$ (siehe (5.3-12)). Für die beiden anderen Verfahren ergibt sich die gesamte zu wiederholende Bediendauer $B^{R_I(n)}$ in Phase I durch Summation über die Anzahl aller Speicherfehler, vgl. (5.2-22), (5.4-16)

$$B^{R_I(n)} = \sum_{1 \le i \le m(I)} q_i^{(n)} \cdot C_i^{(n)} \cdot B_I^{(n)} \cdot D_i^{(n)} / V_I^{(n)}$$
$$\approx \sum_{1 \le i \le m(I)} q_i^{(n)} \cdot a_i \cdot B_I^{(n)}. \qquad (7.1\text{-}7)$$

D.2.2 Bei einem Fehler von Seite s_i ist die Anzahl $\Gamma^{C_i(n)}$ der zur Wiederherstellung der Datenintegrität erneut einzulagernden Seiten gleich der Anzahl der verschiedenen Seiten, auf die während $D_i^{(n)}$ zugegriffen worden ist. Sie ergibt sich durch (5.2-25) bzw. (5.4-17)).
Die Gesamtzahl aller erforderlichen Seiteneinlagerungen Γ^R ergibt sich durch Summation über die Anzahl der entdeckten Fehler (vgl. (5.2-26), (5.3-14), (5.4-18).

$$\Gamma^{R_{I,j}(n)} = \sum_{1 \le i \le m} N^{F_{i,j}(n)} \cdot \Gamma^{C_{i,j}(n)}, \text{ für } j \varepsilon \{1,2\}$$

und

$$\Gamma^{R_{I,j}(n)} = N^{F_j(n)} \cdot \Gamma^{C_j(n)} \text{ , für } j=3$$

Damit erhält man für Phase I die Rate Φ^{HF_I} aller durch Hauptspeicherfehler verursachten zusätzlichen Seiteneinlagerungen:

$$\Phi^{HF_I(n)} = \Gamma^{R_I(n)} / B_i^{(n)} \quad . \qquad (7.1\text{-}8)$$

D.2.3 Nach Abschnitt 5.5 erhält man dann für die Gesamtzahl Γ_x aller erforderlichen Seiteneinlagerungen:

$$\Gamma_x^{(n)} = \Gamma^{R}{}_x^{(n)} + \sum_{1 \le i \le m} C_{i,1}^{(n)},$$

(vgl. (5.5-1) und (5.5-4)). Nach unserem Modell für seitenwechsel-induzierte Fehler (Abschnitt 4.2.2) ist die Gesamt-Paging-Rate Φ^{PF} dann durch (5.5-7) bis (5.5-8) bestimmt:

$$\Phi^{PF}{}_x^{(n)} = \frac{p_P}{(1 - p_P)} \cdot \Gamma_x^{(n)} / B_x^{(n)}$$

$$= \frac{p_P}{(1 - p_P)} \cdot [\Phi^{HF}{}_x^{(n)} + \Phi^{P}{}_x^{(n)}] \; . \qquad (7.1\text{-}9)$$

E. Überprüfung der Konvergenz

Gemäß unserem in Abbildung 7.2 beschriebenen Algorithmus wird jetzt geprüft, ob ein weiterer Iterations-Schritt durchgeführt werden muß, oder ob das **Abruch-Kriterium**

$$|V^{(n)} - V^{(n-1)}| < \varepsilon \qquad (7.1\text{-}10)$$

gültig ist. In diesem Fall terminiert das Iterationsverfahren. Es ist natürlich auch möglich und sinnvoll, andere Bewertungsmaße, wie z.B. die Fehlerwahrscheinlichkeiten q_i, auf Konvergenz zu überprüfen.
In der folgenden Abbildung wollen wir einen abschließenden Überblick geben.

In dieser Abbildung sind die wichtigsten **Gleichungen**, die die Wechselwirkungen zwischen Leistungs- und Fehlermodell abbilden, zusammengefaßt. Um den Überblick nicht zu erschweren, fehlen die Formeln für die vom Verfahren der Fehlerentdeckung abhängigen Größen $D_i^{(n)}$ C_i und $\Gamma^{C}{}_i$. Auf die Ableitung der Fehlerauftritts-Wahrscheinlichkeit q_i wurde an dieser Stelle wegen ihrer Komplexität ebenfalls verzichtet (siehe Teilschritte D.1.2 bis D.1.4). Für eine Fehlerentdeckung durch Konsistenzüberprüfung werden nicht einzelne Seiten s_i betrachtet. Deshalb müssen für dieses Verfahren in den Gleichungen (7.1-6) bis (7.1-8) die Seitenindizes i und die entsprechende Summation unberücksichtigt bleiben (vgl. dazu (5.4-16) bis (5.4-19).

Initialisierung

$B^{R}_{x}{}^{(0)} := 0$, $\Phi^{F}_{x}{}^{(0)} := 0$, $\Phi^{HF}_{x}{}^{(0)} := 0$, $\Phi^{PF}_{x}{}^{(0)} := 0$, $n := 1$

REPEAT

Iterations-Schritt n

C. Spezifiziere das **Leistungsmodell**.

$$B_{x}^{(n)} = \pi_{x} \cdot B + B^{R}_{x}{}^{(n-1)} \qquad (7.1\text{-}1)$$

$$\Phi^{P}_{x}{}^{(n)} = \sum_{i \in I} [\pi_{i}(1-\pi_{i})^{T}\beta_{x} + (1-(1-\pi_{i})^{T})/B_{x}] \qquad (7.1\text{-}2)$$

$$\mu^{CPU}_{x}{}^{(n)} = \Phi^{I/O}_{x} + \Phi^{P}_{x} + \Phi^{HF}_{x}{}^{(n-1)} + \Phi^{PF}_{x}{}^{(n-1)} \qquad (7.1\text{-}3)$$

$$p^{CF}_{x}{}^{(n)} = \frac{\Phi^{I/O}}{\mu^{CPU}_{x}{}^{(n)}}, \quad p^{CP}_{x}{}^{(n)} = 1 - p^{CF}_{x}{}^{(n)}. \qquad (7.1\text{-}4)$$

Bestimme die Verweildauern $V_{x}^{(n)}$ eines Auftrags in einer Lastkonfiguration K.

D. Löse das **Fehlermodell** für Lastkonfiguration K.

Berechne die Verweildauer $D_{i}^{(n)}$ zwischen zwei Fehlerüberprüfungen für die Lastkonfiguration K. Dabei geht nach (7.1-5) das entsprechende Verfahren der Fehlerentdeckung ein und ist eine Funktion f von $V_{x}^{(n)}$.

$$D_{i}^{(n)} = f(V_{x}^{(n)}) \qquad (7.1\text{-}5)$$

Bestimme $q_{i}^{(n)}$, $C_{i}^{(n)}$, $\Gamma^{C}{}_{i}^{(n)}$ und weiterhin

$$\Phi^{F}_{x}{}^{(n)} = \sum_{i \in I} q_{i}^{(n)} \cdot C_{i}^{(n)} / B_{x}^{(n)} \qquad (7.1\text{-}6)$$

$$B^{R}_{x}{}^{(n)} \approx \sum_{i \in I} [q_{i}^{(n)} \cdot a_{i} \cdot B_{x}^{(n)}] \qquad (7.1\text{-}7)$$

$$\Phi^{HF}_{x}{}^{(n)} = \sum_{i \in I} [q_{i}^{(n)} \cdot C_{i}^{(n)} \cdot \Gamma^{C}{}_{i}^{(n)}] / B_{x}^{(n)} \qquad (7.1\text{-}8)$$

$$\Phi^{PF}_{x}{}^{(n)} = \frac{p_{P}}{1 - p_{P}} \cdot (\Phi^{HF}_{x}{}^{(n)} + \Phi^{P}_{x}{}^{(n)}) \qquad (7.1\text{-}9)$$

$n := n+1;$

UNTIL $|V^{(n)} - V^{(n-1)}| > \varepsilon$

Abb. 7.3: Iterations-Algorithmus

F. Integration des Last-Modells mit dem Leistungs- und Fehlermodell

Zur Bestimmung aussagekräftiger Leistungs- und Zuverlässigkeitsmaße müssen die in C und D für eine spezielle Lastkonfiguration $\mathbf{K}=(k_1,..,k_L)$ bestimmten Maße über alle möglichen Lastvektoren gemittelt werden. Man erhält so bspw. für die Verweildauer V_I einer Phase:

$$\bar{V}_I = \sum_{\mathbf{K}} P_{\mathbf{K}} \cdot V_I(\mathbf{K}) \quad . \qquad (7.1\text{-}10)$$

Dabei bezeichnet $V_I(\mathbf{K})$ die für den Lastvektor $\mathbf{K}$ berechnete Verweildauer eines Auftrags der Phase I.
Analog werden Ergebnisse für andere, bspw. die in Abbbildung 7.3 genannten Leistungs- und Zuverlässigkeitsgrößen ermittelt.

7.2 Zur Terminierung der Iteration

Nachdem im letzten Abschnitt das iterative Verfahren zur Modellierung der Wechselwirkungen zwischen Leistung und Zuverlässigkeit vorgestellt worden ist, wollen wir jetzt untersuchen, ob die Iteration auch terminiert.
Dazu interpretieren wir zunächst noch einmal die einzelnen Iterations-Schritte:

Erster Iterations-Schritt:
Im Leistungsmodell des ersten Iterations-Schritts sind noch keine Fehler berücksichtigt, und wir erhalten die Verweildauer $V^{(1)}$ ohne die durch Behandlung von Fehlern verursachten Verzögerungen. Die Bediendauer $B^{(1)}$ ist durch den Benutzer vorgegeben. Sie dient mit der Verweildauer $V^{(1)}$ zur Bestimmung der durch Fehler induzierten zusätzlichen Last.

Zweiter Iterations-Schritt:
Im Leistungsmodell dieses Schritts ist die im ersten Iterations-Schritt bestimmte, durch Fehler induzierte Systemlast enthalten. Diese geht in die Bediendauer $B^{(2)}$ und die Seitenwechsel-Rate $\Phi^{(2)}$ (mit $\Phi=\Phi^{P}+\Phi^{HF}+\Phi^{PF}$) ein. Durch den Last-Zuwachs steigt u.a. die Verweildauer $V^{(2)}$. Mit $V^{(2)}$ und $B^{(2)}$ werden nun $B^{(3)}$ sowie $\Phi^{(3)}$ berechnet.

n-ter Iterations-Schritt:
Für das Leistungsmodell des n-ten Iterations-Schritts ist die Systemlast durch $B^{(n)}$ und $\Phi^{(n)}$ spezifiziert. Damit läßt sich die Verweildauer $V^{(n)}$ bestimmen, und im Fehlermodell die Last $B^{(n+1)}$ und $\Phi^{(n+1)}$ für den nächsten Schritt berechnen.

Definition:
Der **Last-Zuwachs** des n-ten Iterations-Schrittes zum vorhergehenden (n-1)-ten Iterations-Schritt ist durch

$$\sigma B^{(n)} := B^{(n)} - B^{(n-1)} \text{ und } \sigma\Phi^{(n)} := \Phi^{(n)} - \Phi^{(n-1)} \qquad (7.2\text{-}1a)$$

bestimmt. Die **Zunahme der Verweildauer** ist definiert durch:

$$\sigma V^{(n)} := V^{(n)} - V^{(n-1)}. \qquad (7.2\text{-}1b)$$

Für den ersten Iterations-Schritt definieren wir:

$$\sigma B^{(1)} := B,\ \sigma\Phi^{(1)} := \Phi^{P} \text{ und } \sigma V^{(1)} := V\ . \qquad (7.2\text{-}1c)$$

- Offensichtlich sind alle Last-Zuwächse positiv, d.h. die Folgen der $B^{(n)}$, $\Phi^{(n)}$ und $V^{(n)}$ sind monoton steigend. Das Verfahren terminiert dann, wenn die Reihen $\sum_n B^{(n)}$, $\sum_n \Phi^{(n)}$ und damit auch $\sum_n V^{(n)}$ konvergieren.

Wir wollen im folgenden die Einflüsse der verschiedenen Fehler-Arten auf die Terminierung der Iteration getrennt betrachten. Am einfachsten sind dazu Aussagen für Paging-Fehler zu treffen.

Paging-Fehler

Die Paging-Rate Φ^{P} für den fehlerfreien Fall ist durch (6.2-4) bestimmt. Nach (7.1-9) ist die Rate Φ^{PF} der Paging-Fehler durch den Grenzwert einer geometrischen Reihe gegeben:

$$\Phi^{PF} = \frac{p_P}{1 - p_P} \cdot \Phi^{P} \quad ,$$

mit der Wahrscheinlichkeit p_P für eine seitenwechsel-induzierte Störung. Durch Φ^{PF} ist auch die Rate aller durch Paging-Fehler verursachten zusätzlichen Seiteneinlagerungen bestimmt. Die Summenformel für geometrischen Reihen setzt voraus, daß

- $p_P < 1$ (7.2-2)

gilt. In unserem Modell ist normalerweise $p_P \ll 1$, sodaß die Rate Φ^{PF} aller Paging-Fehler sehr schnell konvergiert. Der Zuwachs $\sigma\Phi^{PF(n)}$ im n-ten Iterations-Schritt beträgt:

$$\sigma\Phi^{PF(n)} = \Phi^{P} \cdot p_P^{\,n-1} \quad ,$$

d.h. die Rate der Paging-Fehler konvergiert in unserem Modell, wenn die Wahrscheinlichkeit p_P einer seitenwechsel-induzierten Störung kleiner als 1 ist. Diese Forderung ist jedoch stets erfüllt.

Hauptspeicherfehler

Im Vergleich zu Paging-Fehlern ist der Effekt von Hauptspeicherfehlern auf die Terminierung der Iteration nur schwer abzuschätzen, denn einerseits hängt deren Auftrittsrate von der Auftrags-Verweildauer ab, und andererseits führt die Behandlung dieser Fehler-Art wiederum zu einer Verlängerung der Verweildauer.

Wird die durch Fehler induzierte Last sehr groß, insbesondere größer als die ursprünglich vorhandene Last, kann ein dem **thrashing** vergleichbarer Effekt eintreten, und es wird kein Auftragsfortschritt

erzielt, sondern nur noch Fehlerbehandlung durchgeführt. Die Last-Zuwächse sind dann monoton wachsend und die Iteration terminiert nicht.

Heuristisch läßt sich eine Konvergenz der Reihen $\Sigma\, B^{(n)}$, $\Sigma\, \Phi^{(n)}$ und damit auch von $\Sigma\, V^{(n)}$ wie folgt begründen:

- Im **ersten Schritt** wird die durch Fehler induzierte Last mit Hilfe von $\sigma B^{(1)} := B$ und $\sigma V^{(1)} := V$ bestimmt. Fehler sind sehr seltene Ereignisse, die einen einzelnen Auftrag nur mit sehr geringer Wahrscheinlichkeit betreffen. Deshalb ist der durch Fehler verursachte **mittlere** Last-Zuwachs, d.h. $\sigma B^{(2)}$ und $\sigma\Phi^{(2)}$ meist um Größenordnungen kleiner als die ursprünglich vorhandene, durch B und Φ^P festgelegte Last. Also gilt normalerweise:

  $$\sigma B^{(2)} \ll \sigma B^{(1)} = B \quad \text{und} \quad \sigma\Phi^{(2)} \ll \sigma\Phi^{(1)} = \Phi^P \ .$$

  Weil der durch Fehler induzierte Last-Zuwachs nur gering ist, erhöht sich auch die Verweildauer im Vergleich zur ursprünglichen (im fehlerfreien Fall erhaltenen) Verweildauer nur wenig, d.h. es gilt

  $$\sigma V^{(2)} \ll \sigma V^{(1)} = V.$$

- Jeder **weitere Iterations-Schritt** unterscheidet sich vom vorhergehenden Schritt durch diesen Last-Zuwachs. Auch dieser Last-Zuwachs ($\sigma B^{(n)}$, $\sigma V^{(n)}$) ist Fehlern ausgesetzt und induziert einen nochmals um Größenordnung geringeren Last-Zwachs $\sigma B^{(n+1)}$, $\sigma V^{(n+1)}$, sodaß insgesamt die Last-Zuwächse sehr schnell (exponential) gegen 0 konvergieren, und damit das Iterations-Verfahren terminiert.

Im folgenden wollen wir uns die in den einzelnen Iterations-Schritten induzierte Last noch etwas genauer anschauen, wobei wir zunächst die Auftritts-Wahrscheinlichkeiten von Fehlern betrachten.

Auftritts-Wahrscheinlichkeit von Störungen

Die **Wahrscheinlichkeit** $q^{z_i(n)}$ **einer zugriffsinduzierten Störung** ist für den n-ten Iterations-Schritt durch Gleichung (5.2-16) gegeben:

$$q^{z_i(n)} = 1 - (1 - p_z)^{Z^{c_i(n)}}. \qquad (7.2\text{-}3)$$

Für das erste Verfahren, der **Fehlerentdeckung beim Seiteneinlagern**, ist $Z^{C_i(n)}$ nach (5.2-13) durch

$$Z^{C_i(n)} = \frac{\pi_i \cdot \beta \cdot [1-(1-\pi_i)^T]}{\pi_i (1-\pi_i)^T \cdot \beta + \{[1-(1-\pi_i)^T]/B^{(n)}\}}$$

bestimmt. Weil die Folge der $B^{(n)}$ monoton wächst, ist $Z^{C_i(n)}$ durch

$$Z^{C_i(max)} = \frac{\pi_i \cdot \beta \cdot [1-(1-\pi_i)^T]}{\pi_i (1-\pi_i)^T \cdot \beta + \{[1-(1-\pi_i)^T]/B^{(1)}\}}$$

nach oben beschränkt. Damit konvergiert auch q^{Z_i} gegen einen Wert kleiner als 1.

Für das zweite Verfahren, einer **Fehlerentdeckung beim Seitenzugriff**, gilt $Z^{C_i}=1$, d.h. q^{Z_i} ist konstant mit $q^{Z_i} = p_Z$ (vgl. auch (5.3-7)).

Für das dritte Verfahren, einer **Fehlerentdeckung durch Konsistenzüberprüfung**, ist mit (5.4-7) $Z^{C_i} = B^K \cdot \beta =$ const und damit auch q^{Z_i} = const.

D.h. für alle drei Verfahren konvergieren die q^{Z_i} gegen einen Wert, der kleiner als 1 ist.

Die **Auftrittswahrscheinlichkeit** $q^{U_i(n)}$ **einer ereignisunabhängigen Störung** hängt von der Verweildauer $D_i^{(n)}$ einer Seite s_i im Speicher ab:

$$q^{U_i(n)} = \S(D_i^{(n)}) \quad . \qquad (7.2\text{-}4)$$

Die Funktion §( ) ist durch das Verfahren der Fehlerentdeckung bestimmt, die Verweildauer $D_i^{(n)}$ einer einzelnen Seite ist direkt proportional zur im n-ten Iterations-Schritt bestimmten Verweildauer $V^{(n)}$ eines Auftrags. Die Folge der $q^{U_i(n)}$ ist daher mit $V^{(n)}$ monoton wachsend, und konvergiert nur dann gegen einen Wert kleiner als 1, wenn $V^{(n)}$ beschränkt bleibt.
Die Beschränktheit von V kann nach der oben angegebenen heuristischen Argumentation angenommen werden, wenn die durch Fehler induzierte Last beschränkt bleibt. In "pathologischen" Fällen mit sehr hohen Fehlerwahrscheinlichkeiten kann dies jedoch nicht gewährleistet sein, d.h. in diesen Fällen terminiert die Iteration nicht. Üblicherweise sind die Auftritts-Wahrscheinlichkeiten für Störungen aber sehr klein, so daß es bei praxisnahen Parametern keine Probleme geben sollte.

Konvergiert die monoton wachsende Folge $q_i^{(n)}$ gegen einen Wert q_i^*, dann läßt sich die Beschränktheit der durch Fehler induzierten Last wie folgt nachweisen:

Fehler-induzierte Last

Ein quantitative Größe für die durch Fehler induzierte Last ist der Mittelwert B_I^R der zu wiederholenden Bediendauer.

Für die **Fehlerentdeckung beim Seiteneinlagern** gilt mit $q_i^{(n)}$ nach (5.2-21), (5.2-22):

$$B^R{}_I^{(n)} = \sum_{i \in I} q_i^{(n)} \cdot C_i^{(n)} \; D^{E_i^{(n)} B_I^{(n)}} / V^{(n)}$$

$$= \sum_{i \in I} q_i^{(n)} \cdot \frac{C_i^{(n)}}{C_i^{(n)} + A_i^{(n)}} \; a_i \cdot B_I^{(n)} \leq \Big[\sum_{i \in I} q_i^* \Big] \cdot B_I^{(n)} .$$

Mit $B_I^{(n)} = B_I + B^R{}_I^{(n-1)}$ und $B^R{}_I^{(0)} = 0$ folgt:

$$B^R{}_I := \lim_{n \to \infty} B^R{}_I^{(n)} \leq \sum_{h=1}^{\infty} \Big[\sum_{i \in I} q_i^* \Big]^h \cdot B_I \; .$$

Man erkennt, daß $B^R{}_I$ dann nach oben beschränkt ist, wenn die Summe der Auftritts-Wahrscheinlichkeiten Σq_i^* kleiner als 1 bleibt. Weil Fehler sehr selten vorkommen, gilt normalerweise sogar

- $\sum_{i \in I} q_i^* \ll 1$. (hinreichende Bedingung) (7.2-5)

In den Fällen, in denen die Ungleichung (7.2-5) verletzt ist, kann die zu wiederholende Bediendauer unbegrenzt wachsen und das Iterationsverfahren stoppt nicht. Dies spiegelt genau die Situation wider, daß der Aufwand der Fehlerbehandlung größer als die ursprünglich vorhandene Last wird. Dieser Fall dürfte jedoch bei realistischen Modell-Parametern kaum vorkommen. Dort ist vielmehr die durch Fehler induzierte Last um Größenordnungen kleiner als die ursprüngliche.

Für die **Fehlerentdeckung beim Zugriff** ist $B^R{}_I = 0$ (5.3-13), und die Bediendauer bleibt für jeden Iterations-Schritt konstant.

Für die **Fehlerentdeckung durch Konsistenzüberprüfungen** gilt (vgl. 5.4-16):

$$B^R{}_I^{(n)} = q^{(n)} \cdot B_I^{(n)} .$$

Mit $B_I^{(1)} = B$ folgt:

$$B^R{}_I = \lim_{n \to \infty} B^R{}_I^{(n)} \leq \sum_{h=1}^{\infty} (q^*)^h \cdot B_I \; . \qquad (7.2\text{-}6)$$

- Die geometrische Reihe $B^R{}_I$ konvergiert also, wenn $q^* < 1$ gilt.

Neben der zu wiederholenden Bediendauer wird die Last durch die Rate der **zusätzlich einzulagernden Seiten** Φ^{HF} bestimmt:

$$\Phi^{HF}{}_I{}^{(n)} = [\ \sum_{i \in I} q_i{}^{(n)} \cdot C_i{}^{(n)} \cdot \Gamma^C{}_i{}^{(n)}\ ] / B_I{}^{(n)}\ .$$

Die Zahl der auszulagernden Seiten $\Gamma^C{}_i$ ist stets kleiner als m(I), die Anzahl virtueller Seiten der Phase I. Für das zweite Verfahren einer Fehlerentdeckung beim Zugriff gilt sogar $\Gamma^C{}_i=1$.

Somit kann $\Phi^{HF}{}_I{}^{(n)}$ durch

$$\Phi^{HF}{}_I{}^{(n)} \leq \frac{m(I)}{B_I{}^{(n)}} \cdot \sum_{i \in I} q_i{}^{(n)} \cdot C_i{}^{(n)}$$

nach oben abgeschätzt werden. Wenn nach unserer Voraussetzung die Folge der $q_i{}^{(n)}$ gegen einen Wert kleiner als 1 konvergiert, so konvergieren damit auch $B_I{}^{(n)}$, $C_i{}^{(n)}$ (gegen einen Wert größer als 0), und somit ist $\Phi^{HF}{}_I{}^{(n)}$ nach oben beschränkt.

Abschließend läßt sich feststellen, daß die Iteration um so schneller terminiert, je kleiner die Fehlerauftritts-Wahrscheinlichkeiten q_i sind.

7.3 Auftrags-spezifische Modell-Analyse

In unserem Iterations-Verfahren wurden die Auswirkungen der Fehler auf das Leistungsverhalten (z.B. die Verweildauer) bestimmt, indem wir die durch Fehler induzierte Last für den **stationären Fall - gemittelt über alle Aufträge** - berechnet haben. D.h. es werden die durch die mittlere Anzahl der aufgetretenen Fehler verursachten Verlängerungen der Bedien- und der Verweildauer berechnet. Ein entsprechendes Bewertungs-Maß ist die **apparent capacity** /CASI 80/, die das Verhältnis der tatsächlichen mittleren Verweilzeit zu derjenigen im fehlerfreien System angibt.
In diesem Abschnitt wollen wir jedoch nicht die mittleren, normalerweise relativ geringen Auswirkungen von Fehlern auf die gesamte Systemleistung, sondern diejenigen auf **einen einzelnen Auftrag** betrachten. Wir interessieren uns also für

- die Wahrscheinlichkeit, mit der ein einzelner Auftrag in Abhängigkeit von seiner Bedien- und Verweildauer einen Speicherfehler erleidet,

- die Auswirkung, die ein Fehler auf die Bedien- und Verweildauer eines einzelnen Auftrags besitzt.

Die üblichen, für Mehrprozessor-Systeme definierten Performability-Maße /KHMA 86/ können hier nicht benutzt werden, denn bei diesen wird stets zugrunde gelegt, daß sich die Systemstruktur aufgrund der Einwirkung von Fehlern ändert (structure state model /SMTR 88/). In unserer Modellsicht wird jedoch vornehmlich von einer durch Fehler verursachten Laständerung ausgegangen.

Zeitabhängige Störungsfreiheit eines Auftrags

In unserem Störungsmodell aus Kapitel 4 wird die Wahrscheinlichkeit einer Auftragsstörung durch zwei verschiedene Zeiten beeinflußt: die Bediendauer t_B des Auftrags und seine Verweildauer t_V im Hauptspeicher. Seien X^Z, X^P und X^U Zufallsvariablen für die Zeitdauern, in denen keine zugriffsinduzierten, seitenwechsel-induzierten, bzw. ereignisunabhängigen Hauptspeicherfehler auftreten. Dann definieren wir für jede Fehler-Art mit

$$R^Y(t) = Prob[X^Y > t] \quad , \; Y \varepsilon \{Z,P,U\} \qquad (7.3\text{-}1)$$

die zeitabhängige **Störungsfreiheit eines Auftrags.**

Einfluß der Bediendauer

Durch die Bediendauer t_B eines Auftrags wird die Anzahl der stattgefunden Seitenzugriffe, sowie der regulären Seitenwechsel beeinflußt. D.h. sämtliche ereignisabhängigen Störungen sind in unserem Modell durch die Bediendauer bestimmt.

Zugriffsinduzierte Störungen:
Innerhalb der Bedienzeit t_B eines Auftrags sind im Mittel $\beta \cdot t_B$ Seitenzugriffe erfolgt. Die Wahrscheinlichkeit, mit der bis zum Zeitpunkt t_B keine zugriffsinduzierte Störung erfolgte, erhält man mit Gleichung (5.2-16):

$$R^Z(t_B) := \text{Prob}[X^Z > t_B]$$
$$= (1 - p_Z)^{\beta \cdot t_B}. \qquad (7.3\text{-}2)$$

Seitenwechsel-induzierte Störungen:
Analog erhält man mit der Anzahl der Seitenwechsel nach (5.2-4) für die Wahrscheinlichkeit, daß ein Auftrag keinen Paging-Fehler erleidet:

$$R^P(t_B) := \text{Prob}[X^P > t_B]$$
$$= (1 - p_P)^{\overline{C}(t_B)},$$

mit

$$C(t_B) = \sum_{1 \le i \le m(I)} [1-(1-\pi_i)^T] + \pi_i(1-\pi_i)^T \cdot \beta \cdot t_B$$
$$= \overline{w}(T) + \left[\sum_{1 \le i \le m(I)} \pi_i(1-\pi_i)^T\right] \cdot \beta \cdot t_B \ . \qquad (7.3\text{-}3)$$

Man sieht, daß die Funktion $C(t_B)$ linear mit t_B wächst.

Einfluß der Verweildauer

Ereignisunabhängige Störungen hängen nach dem Modell aus Kapitel 4 von der Zeit t_V ab, die sich der Auftrag im Hauptspeicher befindet. Nach unserem Modell erfährt der Speicher gemäß der Verteilung $F_S(t_V)$ eine ereignisunabhängige Störung (vgl. (4.1-1) bis (4.1-11)). Für die Wahrscheinlichkeit $R^U(t_V)$, daß bis zum Zeitpunkt t_V keine Seite des Hauptspeichers eine ereignisunabhängige Störung erfährt, gilt nach (4.1-7) mit der Verteilung $F_{SVR}(t)$ der Vorwärtsrekurrenzzeit:

$$R^U(t_V) = 1 - F_{SVR}(t_V) \ . \qquad (7.3\text{-}4)$$

Die Wahrscheinlichkeit, mit der eine einzelne Hauptspeicherseite nicht betroffen ist, ergibt sich dann unter Berücksichtigung der Auswirkung von Störungen mit (4.1-19) zu:

$$R^{S}(t_V) = 1 - \sum_{1 \le k \le H} \frac{e(k)}{H} \cdot F_{SVR}(t_V)$$

$$= 1 - p_s \cdot F_{SVR}(t_V) \ , \qquad (7.3\text{-}5)$$

mit p_s nach (5.2-9). Damit läßt sich nun mit der Formel von Poincaré-Sylvester die Wahrscheinlichkeit $R^{A}(t_V)$ errechnen, mit der keine Seite der Arbeitsmenge eines Auftrags durch eine ereignisunabhängige Störung verfälscht wird:

$$R^{A}(t_V) = 1 - \sum_{1 \le k \le T} \{ g(k,T,m) \sum_{1 \le j \le k} (-1)^{j-1} \binom{k}{j} [p_s \cdot F_{SVR}(t_V)]^j \} \qquad (7.3\text{-}6)$$

Dabei bezeichnet g(k,T,m) die Verteilung der Arbeitsmengengröße nach (6.2-4). Setzt man voraus, daß es sehr unwahrscheinlich ist, daß eine Seite aus der Arbeitsmenge von mehreren Störungen betroffen ist, d.h. daß für $j \geq 2$:
$[p_s(1 - F_{SVR}(t_V))]^j \approx 0$ gilt, läßt sich (7.3-6) durch:

$$R^{A}(t_V) \approx 1 - \bar{w}(T) \cdot (p_s \, F_{SVR}(t_V)) \qquad (7.3\text{-}7)$$

abschätzen. An dieser Gleichung lassen sich gut die relevanten Einflußgrößen ablesen. Beim Subtrahend steht der erste Faktor w(T) für die mittlere Größe des vom Auftrag belegten Speicherplatzes. Der geklammerte Faktor ist eine Funktion der vom Auftrag im Speicher zugebrachten Zeit. Die Größe p_s beschreibt dabei die von der Hardware-Architektur des Speichers abhängige Auswirkung einer Störung. Das Risiko einer Auftrags-Störung wächst also mit dem **space time product** /GEMI 80/.

Für den in der Praxis viel benutzten Poissonprozeß der Störungen ergibt sich aus (7.3-7):

$$R^{A}(t_V) \approx 1 - \bar{w}(T) \cdot p_s \cdot (1 - e^{-\delta_s \cdot t_V}) \ . \qquad (7.3\text{-}8)$$

Als Störungsfreiheit $R(t_B,t_V)$ eines Auftrags für alle Störungs-Arten erhält man dann insgesamt:

$$R(t_B,t_V) = R^{Z}(t_B) \cdot R^{P}(t_B) \cdot R^{A}(t_V) \ , \qquad (7.3\text{-}9)$$

wenn - wie in unserem Modell vorausgesetzt - alle Störungs-Arten stochastisch unabhängig voneinander auftreten. Man erkennt an der Notation, daß die Störungsfreiheit $R(t_B,t_V)$ eines Auftrags von den beiden Zeitdauern t_B und t_V abhängt.

Die Beziehung zwischen der Bedien- und der Verweildauer wird im Leistungsmodell (Kapitel 6) durch ein Warteschlangen-Netz abgebildet. Sie ist von der Systemlast abhängig; eine geschlossene Form existiert dafür nicht.

Auswirkung eines entdeckten Fehlers

Wie bereits in Kapitel 5 beschrieben, wirkt sich ein Fehler auf die Bediendauer, die Paging-Rate sowie die Verweildauer aus.

Verlängerung der Bediendauer

In Gleichung (7.1-7) wird die Verlängerung $\overline{B}^R$ der Bediendauer berechnet, die ein Auftrag im Mittel durch die erlittenen Fehler erfährt. Weil die mittlere Fehler-Anzahl eines Auftrags normalerweise sehr viel kleiner als 1 ist, ist die mittlere Verlängerung der Bediendauer auch um Größenordnungen kleiner als diejenige, die ein Auftrag wirklich im Fehlerfall erfährt. Wird bei einer Seite s_i ein Fehler entdeckt, dann ist die entsprechende Verlängerung $\overline{B}^{R_i}$ für eine Fehlerentdeckung beim Seitenauslagern nach Gleichung (5.2-22):

$$\overline{B}^{R_i} = \frac{\overline{D}^{E_i} \cdot B_I}{\overline{V}_I} \quad .$$

Eine Fehlerentdeckung durch Konsistenzüberprüfung ergibt sich als Spezialfall dieser Gleichung ($B^{R_i} = B^K$, für alle i). Für die **Verteilung $F_B(t)$ der Bediendauer** läßt sich dann für einen gegebenen Auftrag (mit der Bedienanforderung B_I und der Verweildauer V_I) die folgende rekursive Formel angeben (eine ähnliche Fragestellung für ein allgemeines rollback wird in /HEID 83/ und /SCHN 83/ behandelt):

$$F_B(t) = [R^Z(B_I) \cdot R^A(V_I)] \cdot h(t-B_I) + [1 - R^Z(B_I) \cdot R^A(V_I)] \cdot \sum_{i \in I} \frac{a_i}{\overline{w}_I(T)} F_B(t-\overline{B}^{R_i}) \quad , \qquad (7.3\text{-}10)$$

wobei h(t) die heavyside-Funktion ist mit $h(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}$

Gleichung (7.3-10) kann folgendermaßen interpretiert werden:

Der erste Summand behandelt nach dem Gesetz von der totalen Wahrscheinlichkeit den fehlerfreien Fall, in dem die Bediendauer genau gleich B_I ist. Hierbei ist zu beachten, daß nur ereignisunabhängige und zugriffsinduzierte Störungen zu berücksichtigen sind, denn nur sie führen zu einer Verlängerung der Bediendauer.

Der zweite Summand modelliert den Fehlerfall, der mit der Wahrscheinlichkeit $[1 - R^{z}(B_I) \cdot R^{\wedge}(V_I)]$ eintritt. Dann ist eine bestimmte virtuelle Seite s_i mit der Wahrscheinlichkeit $a_i/\overline{w}_I(T)$ betroffen, denn die mittlere Arbeitsmengengröße $\overline{w}_I(T)$ ist gerade die Summe aller Aufenthaltswahrscheinlichkeiten a_i, vgl. (6.2-3). Die Bediendauer erhöht sich dann gerade um die Zeitdauer B^{R_i}.
Leider kann keine geschlossene Form angegeben werden, aber z.B. über die Laplacetransformation /DOET 81/ können die Momente der Bediendauer-Verteilung errechnet werden.

Erhöhung der Paging-Rate und der Verweildauer:

Jede Fehlerentdeckung bei einer beliebigen, aber festen Seite s_i aus der Phase I führt demnach zu einem zusätzlichen Auftrag, der

- die Bedienanforderung $\overline{B}^{R_i}$ besitzt und
- eine zusätzliche Anzahl von $\overline{\Gamma}^{C_i}$ Seiten-Einlagerungen (vgl. (5.2-26) und (5.4-17)) erfordert.

Der durch diesen Fehler erzeugte Auftrag ist demnach durch die Paging-Rate:

$$\Phi^{PR_i} = \overline{\Gamma}^{C_i}/\overline{B}^{R_i} + \Phi^{P} \qquad (7.3\text{-}11)$$

spezifiziert. Der erste Summand enthält die zur Wiederherstellung der Datenintegrität erforderlichen Einlagerungen, der zweite die bei der fehlerfreien Auftragsbearbeitung auftretenden Seitenwechsel.

Die von einem bestimmten (Teil-)**Wiederholungsauftrag** benötigte **Verweildauer** $\overline{V}^{R_i}$ kann nicht in geschlossener Form berechnet werden, sie ist u.a. von der aktuellen Systemlast abhängig. Man erhält sie, indem man einen einzelnen, durch $\overline{B}^{R_i}$ und Φ^{PR_i} spezifizierten Auftrag in das Warteschlangen-Netz aus Kapitel 6 hinzufügt. Die zur Bearbeitung dieses Auftrags erforderliche Verweildauer ist dann die gesuchte Größe.
Eine Beispiel-Berechnung wird dazu in Kapitel 8 durchgeführt.

8 Analyse anhand von Beispielen

Im ersten Abschnitt dieses Kapitels führen wir das iterative Lösungsverfahren aus Abschnitt 7.1 an einem konkreten Beispiel durch. Im zweiten Abschnitt sollen dann anhand dieses Beispiels die Auswirkungen einiger Modell-Parameter auf die Leistung und die Zuverlässigkeit genauer untersucht werden.

8.1 Durchführung der Iteration

Die erforderlichen Berechnungen, sowie die eigentliche Iteration wurden in Turbo-PASCAL auf einem IBM-AT Personal Computer implementiert.
Zur Bestimmung der Leistungsgrößen des Warteschlangen-Netzwerkes wurden die Algorithmen der **mean value analysis** /LAVE 83/, /REIS 79/, /RELA 80/ benutzt.
Wir wollen nun die Schritte A bis D aus dem vorhergehenden Kapitel durchführen. Die gewählten Größen der Parameter orientierten sich an anderen Modellbeispielen aus der Literatur /COUR 77/, /BARD 77/, /HEMO 81/.

A. Spezifikation des Modells

Der **Auftragsfluß** sei spezifiziert durch:

- gesamte Bedienzeitanforderung: $B^G = 10$ s,
- Seiten-Zugriffsrate: ß = 250 Zugriffe s^{-1},
- Bedienrate für den I/O-Transfer: $\mu^{I/O} = 2\ s^{-1}$,
- Bedienrate fürs Paging: $\mu^P = 2\ s^{-1}$;
- Rate für einen File Transfer: $\Phi^{I/O} = 25$ Tansfers$\cdot s^{-1}$.

Für die **Speicherverwaltung** sollen die folgenden Voraussetzungen gelten:

- Speichergröße: H = 64 Seiten,
- Größe einer Seite: 8 Kbyte,
- Anzahl der virtuellen Seiten eines Auftrags: M = 40 ,
- Fenstergröße: T = 8.

Der Zugriff auf die virtuellen Speicherseiten werde durch die Zugriffsmatrix Q spezifiziert, die NCD-Form besitzt und sich wie folgt in drei quadratische Submatrizen zerlegen läßt:

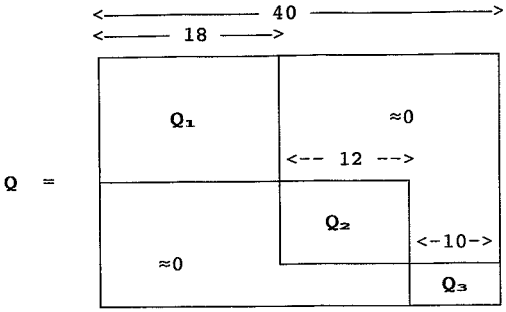


Abb. 8.1: Zerlegung der Übergangsmatrix Q in Submatrizen

Die Matrizen Q_1,Q_2 und Q_3 besitzen die folgenden Werte:

Matrix Q_1:

-0,9708 0,0017 0,0326 0,0258 0,0600 0,0481 0,0378 0,0361 0,0549 0,1511 0,0275 0,0137 0,1562 0,0052 0,0755 0,0979 0,1356 0,0704
0,0441 -0,9746 0,0068 0,0390 0,0068 0,0475 0,0390 0,0119 0,1272 0,0085 0,0475 0,1272 0,0763 0,0695 0,1001 0,1086 0,0780 0,0356
0,0083 0,0083 -0,9967 0,0066 0,0347 0,0413 0,0347 0,0066 0,0991 0,0743 0,0661 0,0182 0,0809 0,1338 0,0446 0,1602 0,1585 0,0198
0,0275 0,0216 0,0511 -0,9548 0,0157 0,0334 0,0354 0,0039 0,1062 0,1790 0,0118 0,0315 0,0138 0,1652 0,1554 0,0197 0,0354 0,0472
0,0034 0,0135 0,0489 0,0017 -0,9899 0,0287 0,0186 0,0017 0,0894 0,1046 0,0759 0,0068 0,0000 0,1384 0,1401 0,0945 0,0574 0,1654
0,0367 0,0230 0,0444 0,0184 0,0321 -0,9740 0,0122 0,0015 0,1499 0,0230 0,0796 0,0796 0,1193 0,0138 0,0673 0,1392 0,1102 0,0230
0,0313 0,0156 0,0156 0,0348 0,0504 0,0469 -0,9896 0,0122 0,0452 0,0782 0,1025 0,1338 0,0434 0,0104 0,0678 0,1407 0,1129 0,0469
0,0292 0,0107 0,0445 0,0399 0,0230 0,0861 0,0246 -0,9693 0,0292 0,1274 0,0476 0,0000 0,0998 0,1258 0,0706 0,0844 0,0706 0,1351
0,0293 0,0264 0,0088 0,0146 0,0396 0,0190 0,0264 0,0029 -0,9648 0,1392 0,0615 0,1128 0,1377 0,0249 0,0718 0,1260 0,0366 0,0864
0,0224 0,0224 0,0266 0,0224 0,0308 0,0084 0,0028 0,0042 0,0840 -0,9188 0,0434 0,0084 0,1035 0,1357 0,0966 0,1091 0,1021 0,0952
0,0139 0,0243 0,0069 0,0468 0,0364 0,0503 0,0434 0,0052 0,0885 0,0763 -0,9775 0,0885 0,0763 0,0746 0,1093 0,0156 0,1336 0,0867
0,0177 0,0197 0,0374 0,0118 0,0000 0,0020 0,0551 0,0374 0,1160 0,0708 0,0079 -0,9705 0,0767 0,1573 0,1200 0,1888 0,0354 0,0157
0,0186 0,0290 0,0352 0,0600 0,0083 0,0600 0,0393 0,0207 0,0248 0,0331 0,0538 0,0103 -0,9214 0,0083 0,1944 0,0476 0,1055 0,1717
0,0019 0,0507 0,0432 0,0188 0,0394 0,0207 0,0169 0,0038 0,0263 0,0751 0,1239 0,0582 0,0413 -0,9080 0,0620 0,1502 0,0056 0,1690
0,0185 0,0370 0,0023 0,0648 0,0532 0,0162 0,0208 0,0601 0,0856 0,1550 0,0532 0,0948 0,0139 0,1203 -0,9561 0,1388 0,0023 0,0185
0,0167 0,0097 0,0306 0,0404 0,0195 0,0209 0,0042 0,0390 0,0320 0,0821 0,1294 0,0292 0,1127 0,0737 0,0821 -0,9165 0,0682 0,1252
0,0200 0,0107 0,0027 0,0173 0,0240 0,0213 0,0173 0,0266 0,1146 0,0653 0,1079 0,0799 0,0946 0,0879 0,0959 0,1066 -0,9347 0,0413
0,0284 0,0213 0,0124 0,0231 0,0302 0,0444 0,0035 0,0053 0,0781 0,1313 0,1650 0,0337 0,0089 0,1278 0,1313 0,0461 0,0124 -0,9042

Matrix Q_2:

-0,9512 0,0029 0,0546 0,0431 0,0000 0,2527 0,1780 0,0029 0,0919 0,2527 0,0459 0,0230
0,0450 -0,9936 0,0086 0,0364 0,0193 0,0879 0,0129 0,2036 0,1800 0,0921 0,1157 0,1886
0,0510 0,0155 -0,9445 0,0333 0,0621 0,1665 0,0999 0,0910 0,1309 0,1420 0,1021 0,0466
0,0170 0,0170 0,0068 -0,9864 0,0714 0,2211 0,0374 0,0816 0,2041 0,1530 0,1360 0,0374
0,0248 0,0028 0,0469 0,0469 -0,9558 0,0331 0,0662 0,2512 0,1546 0,1187 0,0497 0,1573
0,0643 0,0071 0,0143 0,0036 0,0571 -0,9429 0,0250 0,3000 0,2822 0,0357 0,0643 0,0857
0,0061 0,0242 0,0878 0,0030 0,0182 0,1121 -0,8758 0,1242 0,1605 0,1878 0,1363 0,0121
0,0406 0,0244 0,0264 0,0122 0,0284 0,1989 0,1502 -0,8072 0,1400 0,0852 0,0629 0,0345
0,0188 0,0023 0,0422 0,0586 0,0281 0,1219 0,1829 0,0211 -0,8968 0,2134 0,1688 0,0352
0,0380 0,0190 0,0190 0,0422 0,0612 0,0992 0,1182 0,1626 0,0549 -0,9050 0,1246 0,1626
0,0748 0,0479 0,0868 0,0030 0,0748 0,0808 0,0269 0,0509 0,1766 0,1077 -0,8653 0,1317
0,0346 0,0432 0,0195 0,0281 0,0454 0,0000 0,1405 0,1773 0,0994 0,1189 0,0994 -0,8098

Matrix Q_3:

-0.9533 0.0027 0.0522 0.0412 0.1099 0.2417 0.1703 0.0027 0.0879 0.2417
0.0648 -0.9302 0.0523 0.0075 0.1097 0.1421 0.1969 0.1022 0.0150 0.2368
0.0104 0.0600 -0.9896 0.0731 0.1644 0.2010 0.1957 0.0130 0.0731 0.1957
0.0652 0.0026 0.0757 -0.9374 0.1201 0.0548 0.2479 0.1696 0.0313 0.1670
0.0586 0.0698 0.0586 0.0112 -0.8324 0.1257 0.1117 0.0307 0.1368 0.2262
0.0497 0.0497 0.0468 0.0058 0.0702 -0.7339 0.1637 0.1257 0.0526 0.1667
0.0791 0.0088 0.0176 0.0044 0.0264 0.0703 -0.9693 0.3689 0.3470 0.0439
0.0698 0.0931 0.0078 0.0310 0.0737 0.2366 0.0233 -0.8565 0.1591 0.1591
0.0077 0.0052 0.0129 0.0618 0.0000 0.2113 0.2138 0.1443 -0.9124 0.2525
0.0798 0.0498 0.0964 0.0399 0.1030 0.0565 0.0598 0.1363 0.3257 -0.9501

Die Wahrscheinlichkeiten eines Phasenwechsels werden als konstant angenommen, d.h. wir setzen die **lumpability**-Eigenschaft (siehe (6.1-9)) voraus. Das Phasenwechsel-Verhalten werde durch die folgende Matrix **Q** der aggregierten Zustände bestimmt. Die Werte in dieser Matrix entsprechen den Übergangsraten, mit denen ein Wechsel von einem Zustand der Submatrix Q_I in einen Zustand einer anderen Submatrix Q_J erfolgt.

Q =

Q_1	.0002	.0007
.0030	Q_2	.0005
.0025	.0005	Q_3

Abb. 8.2: Matrix zur Spezifikation des Phasen-Wechsels

Zur vollständigen Modell-Spezifikation muß noch das Auftreten von Störungen festgelegt werden.

Dazu nehmen wir in unserem Beispiel an, daß der zeitliche Abstand ereignisunabhängiger Störungen **exponential verteilt** ist, d.h.:

- $F_S(t) = 1 - e^{-\delta_S \cdot t}$ mit $\delta_S = 0.002\ s^{-1}$.

Nach unserem Modell zur Auswirkung von Störungen sollen von einer einzelnen Störung im Mittel

- $\bar{e} = 6.4$

Seiten betroffen sein.

Ereignisabhängige Störungen sollen mit der Wahrscheinlichkeit :

- $p_z = 0.001$ durch Zugriffe
- $p_p = 0.010$ durch Seitenwechsel

induziert werden.

B. Lastmodell

B.1 Anzahl der Phasen

Die Anzahl der Speicher-Phasen eines Auftrags ergibt sich aus der Zahl der quadratischen Submatrizen, in die sich die Matrix $\mathbf{Q}$ aus Abbildung 8.1 zerlegen läßt. D.h. in unserem Beispiel gilt L = 3.

B.2 Phasenverhalten

Im nächsten Schritt unserer Lastmodellierung müssen die **stationären Zugriffswahrscheinlichkeiten** aller virtuellen Seiten bestimmt werden. Dazu muß die Matrix $\mathbf{Q}$ nach (6.1-8) in Blockdiagonalform $\mathbf{Q}^*$ transformiert werden.
Mit Hilfe der **Power-Methode** /STEW 78/, /BOAK 82/ werden dann für diese Matrizen die stationären Zustandswahrscheinlichkeits-Vektoren π_I^*, I=1,2,3, durch Lösung der Gleichungen $\pi_I^* = Q_I^* \cdot \pi_I^*$ bestimmt. Es ergibt sich:

Phase 1: m(1) = 18
$\pi_1^* = (\pi_{1(1)}^*, \pi_{2(1)}^*, \ldots, \pi_{18(1)}^*)$:
[0.020 0.024 0.023 0.032 0.028 0.028 0.022 0.019 0.071 0.094 0.076 0.053 0.072 0.090 0.097 0.100 0.062 0.089]

Phase 2: m(2) = 12
$\pi_2^* = (\pi_{1(2)}^*, \pi_{2(2)}^*, \ldots, \pi_{12(2)}^*)$:
[0.039 0.020 0.042 0.025 0.043 0.112 0.110 0.141 0.144 0.129 0.109 0.087]

Phase 3: m(3) = 10
$\pi_3^* = (\pi_{1(3)}^*, \pi_{2(3)}^*, \ldots, \pi_{10(3)}^*)$:
[0.054 0.043 0.041 0.031 0.076 0.170 0.121 0.145 0.160 0.159]

Die Wahrscheinlichkeiten $\pi_{i(I)}^*$ spezifizieren für jede einzelne Phase ein independent reference model (IRM), siehe Abschnitt 2.1.

Als nächstes müssen die marginalen Wahrscheinlichkeiten π_I, I=1,2,3 , berechnet werden, mit denen sich ein Auftrag in einer bestimmten Phase befindet. Dazu wird der Phasenwechsel durch die folgende Matrix $\mathbf{P}$ beschrieben:

$$\mathbf{P} = \begin{bmatrix} -0.0009 & 0.0002 & 0.0007 \\ 0.0030 & -0.0035 & 0.0005 \\ 0.0025 & 0.0005 & -0.0030 \end{bmatrix}$$

Mit Hilfe der Power-Methode läßt sich $\boldsymbol{\pi} = \mathbf{P} \cdot \boldsymbol{\pi}$ lösen, und man erhält für den Vektor $\boldsymbol{\pi}=(\pi_1,\pi_2,\pi_3)$:

$\pi_1 = 0.625$, $\pi_2 = 0.151$, $\pi_3 = 0.224$.

B.3 Lastverteilung

Für jede Phase I wird mit (6.2-5) die mittlere Arbeitsmengen-Größe $w_I(T)$ bestimmt. Bei einer Fenstergröße von T=8 ergeben sich:

$\overline{w}_1(T) = 6.28$, $\overline{w}_2(T) = 5.62$, $\overline{w}_3(T) = 5.27$.

Befinden sich K Aufträge im Dispatching-System, dann gibt es nach (6.1-5)

$$|\mathbf{K}| = \binom{K+2}{K}$$

verschiedene Lastvektoren $\mathbf{K}=(k_1,k_2,k_3,)$ ohne Berücksichtigung der Randbedingung (6.1-2). Die Anzahl $|\mathbf{K}|$ der Lastvektoren für verschiedene Auftrags-Anzahlen K ist in folgender Tabelle dargestellt:

K	1	2	3	4	5	6	7	8	9	10	11	12	20
$\|\mathbf{K}\|$	3	6	10	15	21	28	36	45	55	66	78	91	231

Nach dem working-set-Prinzip, bzw. nach Randbedingung (6.1-2) muß $\sum_{1 \le I \le I_*} k_I \cdot w_I(T) \le H$ gelten.

Weil in unserem Beispiel für I=1,2,3 stets:

$5.27 < \vec{w}_I(8) < 6.28$ gilt, können bei H=64 höchstens 12 Aufträge in den Speicher eingelagert werden. Weniger als 10 Aufträge beanspruchen immer weniger als 64 Seiten, sodaß die Speicherbegrenzung nicht zum Tragen kommt. Mit (6.1-4) bis (6.1-7) lassen sich die Auftrittswahrscheinlichkeiten für jeden Lastvektor bestimmen. Die folgende Tabelle 8.1 zeigt für unser Beispiel mit K=10 alle möglichen Lastvektoren, sowie deren Auftrittswahrscheinlichkeiten.

Tabelle 8.1: Lastvektoren mit ihren Auftrittswahrscheinlichkeiten, bei K=10 Aufträgen. Der nötige Speicherplatz ist in Seiten angegeben.

Nr	Lastvektor	$P_{\vec{x}}$	Speicherplatz	Nr	Lastvektor	$P_{\vec{x}}$	Speicherplatz
1	0 0 10	0.00000	52.70	34	3 3 4	0.00889	56.78
2	0 1 9	0.00000	53.05	35	3 4 3	0.00599	57.13
3	0 2 8	0.00001	53.40	36	3 5 2	0.00242	57.48
4	0 3 7	0.00001	53.75	37	3 6 1	0.00054	57.83
5	0 4 6	0.00001	54.10	38	3 7 0	0.00005	58.18
6	0 5 5	0.00001	54.45	39	4 0 6	0.00405	56.74
7	0 6 4	0.00001	54.80	40	4 1 5	0.01637	57.09
8	0 7 3	0.00000	55.15	41	4 2 4	0.02759	57.44
9	0 8 2	0.00000	55.50	42	4 3 3	0.02480	57.79
10	0 9 1	0.00000	55.85	43	4 4 2	0.01254	58.14
11	0 10 0	0.00000	56.20	44	4 5 1	0.00338	58.49
12	1 0 9	0.00001	53.71	45	4 6 0	0.00038	58.84
13	1 1 8	0.00005	54.06	46	5 0 5	0.01335	57.75
14	1 2 7	0.00015	54.41	47	5 1 4	0.04569	58.10
15	1 3 6	0.00023	54.76	48	5 2 3	0.06159	58.45
16	1 4 5	0.00023	55.11	49	5 3 2	0.04152	58.80
17	1 5 4	0.00016	55.46	50	5 4 1	0.01399	59.15
18	1 6 3	0.00007	55.81	51	5 5 0	0.00189	59.50
19	1 7 2	0.00002	56.16	52	6 0 4	0.03151	58.76
20	1 8 1	0.00000	56.51	53	6 1 3	0.08497	59.11
21	1 9 0	0.00000	56.86	54	6 2 2	0.08592	59.46
22	2 0 8	0.00011	54.72	55	6 3 1	0.03861	59.81
23	2 1 7	0.00060	55.07	56	6 4 0	0.00651	60.16
24	2 2 6	0.00142	55.42	57	7 0 3	0.05024	59.77
25	2 3 5	0.00181	55.77	58	7 1 2	0.10161	60.12
26	2 4 4	0.00161	56.12	59	7 2 1	0.06850	60.47
27	2 5 3	0.00087	56.47	60	7 3 0	0.01539	60.82
28	2 6 2	0.00029	56.82	61	8 0 2	0.05257	60.78
29	2 7 1	0.00006	57.17	62	8 1 1	0.07088	61.13
30	2 8 0	0.00000	57.52	63	8 2 0	0.02389	61.48
31	3 0 7	0.00083	55.73	64	9 0 1	0.03260	61.79
32	3 1 6	0.00361	56.08	65	9 1 0	0.02197	62.14
33	3 2 5	0.00791	56.43	66	10 0 0	0.00900	62.80

Befinden sich 11 Aufträge im Dispatching-System, dann sind aufgrund der Randbedingung nur 51 von 78 möglichen Lastvektoren zulässig.

C. Leistungsmodell

Nachdem im Schritt B unter der Voraussetzung, daß K Aufträge aktiviert sind, für alle möglichen Lastvektoren ihre Aufrittswahrscheinlichkeiten bestimmt worden sind, muß im Schritt C für jeden Lastvektor ein einzelnes Leistungsmodell berechnet werden.
In unserem Beispiel wollen wir voraussetzen, daß sich die durch den Lastvektor

- $\mathbf{K} = (6,2,2)$

bestimmte Last im Dispatching-System befindet. Aus Tabelle 8.1 ist ersichtlich, daß dieser Lastvektor mit der Wahrscheinlichkeit $p_{(6,2,2)} = 0.08592$ auftritt.

Die Berechnung von Leistungsgrößen führen wir nach dem Verfahren der **mean value anlysis** /LAVE 83/, /REIS 79/, /RELA 80/ durch.

Erster Iterationsschritt

Im ersten Iterationsschritt werden noch keine Fehler berücksichtigt.

In Schritt A ist eine Gesamt-Bedienanforderung B^G = 10s vorausgesetzt. Mit den Phasenwahrscheinlichkeiten π_I, I=1,2,3, erhält man mit (7.1-1) für die einzelnen Phasen als Bedienanforderungen:

$B_1 = 6.25$, $B_2 = 1.51$, $B_3 = 2.24$.

Anschließend ergeben sich mit (7.1-2) die Paging-Raten:

$\Phi^{P}{}_1 = 147.240$, $\Phi^{P}{}_2 = 111.565$, $\Phi^{P}{}_3 = 95.508$.

Im ersten Iterationsschritt gilt, wie vorausgesetzt

$B^{R}{}_I{}^{(0)} = 0$, $\Phi^{F}{}_I{}^{(0)} := 0$, $\Phi^{HF}{}_I{}^{(0)} := 0$ und $\Phi^{PF}{}_I{}^{(0)} := 0$.

Damit erhalten wir mit (7.1-3) als CPU-Bedienraten:

$\mu^{CPU}{}_1 = 172.24$, $\mu^{CPU}{}_2 = 136.57$, $\mu^{CPU}{}_3 = 120.51$.

In die mean value analysis gehen nicht die Übergangswahrscheinlichkeiten, sondern die relativen Besuchshäufigkeiten e_i ein. Sie werden für jede Lastklasse mit dem linearen Gleichungssystem /BUZE 73/

$$e_i = \sum_{1 \le j \le N} e_j \cdot p_{ji} \quad \text{und} \quad \sum_{1 \le i \le N} e_i = 1 \qquad (8.1\text{-}1)$$

bestimmt, wobei hier N die Gesamtzahl der Stationen und i,j Indizes der Stationen sind. In unserem central server Modell erhalten wir nach Lösung von (8.1-1):

$e_1 = 0.5, \quad e_2 = p^{CF}/2 \quad \text{und} \quad e_3 = p^{CF}/2 \ .$

Numerisch ergibt sich für den ersten Iterationsschritt:

Phase 1:	e_1=0.500	e_2=0.073	e_3=0.427
Phase 2:	e_1=0.500	e_2=0.092	e_3=0.408
Phase 3:	e_1=0.500	e_2=0.104	e_3=0.396.

Nun ist das Leistungsmodell für den ersten Schritt spezifiziert. Die Ergebnisse der mean value analysis sind in Tabelle 8.2 dargestellt.

Tabelle 8.2: **Ergebnisse des Leistungsmodells nach dem ersten Iterationsschritt.**

Gesamt-Durchsatz:		
Phase 1: 13.79	Phase 2: 4.20	Phase 3: 4.00
CPU-Zyklus-Zeit $\bar{T}^{C}_{I}$:		
Phase 1: 0.87	Phase 2: 0.95	Phase 3: 1.00
Anzahl der Zyklen $\bar{N}^{C}_{I}$:		
Phase 1: 1076.50	Phase 2: 206.21	Phase 3: 269.94
Verweildauern im Speicher $\bar{V}_{I}$:		
Phase 1: 936.66	Phase 2: 196.13	Phase 3: 269.35

In Tabelle 8.2 bezeichnet die CPU-Zyklus-Zeit $\bar{T}^{C}_{I}$ die mittlere Verweildauer, die sich ein Auftrag der Phase I zwischen dem Beginn zweier sukzessiver CPU-Bedienungen im Dispatching-System befindet /LAVE 83/. Die mittlere Anzahl $\bar{N}^{C}_{I}$ der für die Bedienung eines Auftrags erforderlichen Zyklen ergibt sich durch :

$$\bar{N}^{C}_{I} = B_I \cdot \mu^{CPU}_{I} \ . \qquad (8.1\text{-}2)$$

Die mittlere Gesamtverweildauer $\bar{V}_I$ eines Auftrags in Phase I ist dann:

$$\bar{V}_I = \bar{T}^{C}_{I} \cdot \bar{N}^{C} \ . \qquad (8.1\text{-}3)$$

Außer der für unser Fehlermodell benötigten Verweildauer können natürlich auch noch andere Leistungsgrößen, wie z.B. Durchsatz, Warteschlangen-Längen und Auslastung an einzelnen Stationen, berechnet werden (siehe z.B. /LAVE 83/).

D. Fehlermodell

Mit den oben berechneten Werten für die Verweildauer eines Auftrags kann nun das Fehlermodell analysiert werden.
Exemplarisch sollen die Ergebnisse des Fehlermodells bei einer **Fehlerentdeckung beim Auslagern** für die virtuellen Seiten der Phase 1 gezeigt werden; vgl. Tabelle 8.3.

Tabelle 8.3: **Ergebnisse des Fehlermodells bei einer Fehlerentdeckung beim Auslagern für Phase 1**

n_1	$\bar{V}_1$	$\bar{C}_1$	$\bar{D}^{R}_1$	q''_1	q^{F}_1	q_1	$\bar{N}^{F}_1$	$\bar{B}^{R}_1$
0.020	139.78	26.74	5.23	0.00106	0.00017	0.00123	0.033	0.0011
0.024	165.44	31.05	5.33	0.00108	0.00021	0.00129	0.040	0.0014
0.023	159.09	30.00	5.30	0.00107	0.00020	0.00127	0.032	0.0014
0.032	214.58	38.77	5.53	0.00112	0.00030	0.00141	0.055	0.0020
0.028	190.36	35.06	5.43	0.00109	0.00025	0.00135	0.047	0.0017
0.028	190.36	35.06	5.43	0.00109	0.00025	0.00135	0.047	0.0017
0.022	152.70	28.93	5.28	0.00106	0.00019	0.00126	0.036	0.0013
0.019	133.26	25.61	5.20	0.00105	0.00017	0.00121	0.031	0.0011
0.071	417.01	61.99	6.73	0.00136	0.00080	0.00216	0.134	0.0060
0.094	511.44	67.22	7.61	0.00154	0.00119	0.00273	0.184	0.0093
0.076	438.97	63.57	6.91	0.00140	0.00088	0.00227	0.144	0.0067
0.053	330.79	53.92	6.13	0.00124	0.00054	0.00178	0.096	0.0039
0.072	421.47	62.33	6.76	0.00137	0.00081	0.00218	0.136	0.0061
0.090	496.19	66.66	7.44	0.00151	0.00112	0.00262	0.175	0.0087
0.097	522.58	67.56	7.73	0.00157	0.00125	0.00282	0.190	0.0098
0.100	533.46	67.83	7.86	0.00159	0.00131	0.00290	0.197	0.0103
0.062	375.35	58.46	6.42	0.00130	0.00066	0.00196	0.115	0.0049
0.089	492.30	66.50	7.40	0.00150	0.00110	0.00260	0.173	0.0085

Die Ergebnisse für die einzelnen virtuellen Seite müssen anschließend nach den Gleichungen (7.1-5) bis (7.1-9) zusammengefaßt werden. Wir erhalten die in Tabelle 8.4 zusammengestellten Werte:

Tabelle 8.4: **Fehler-Raten und Verlängerung der Bediendauern für die verschiedenen Phasen**

Phase 1:	$\Phi^{F}_1 = 0.300$	$\Phi^{HF}_1 = 0.795$	$\Phi^{PF}_1 = 1.49$	$B^{R}_1 = 0.086$
Phase 2:	$\Phi^{F}_2 = 0.292$	$\Phi^{HF}_2 = 0.502$	$\Phi^{PF}_2 = 1.13$	$B^{R}_2 = 0.024$
Phase 3:	$\Phi^{F}_3 = 0.288$	$\Phi^{HF}_3 = 0.664$	$\Phi^{PF}_3 = 0.96$	$B^{R}_3 = 0.042$

In Tabelle 8.4 sind in den Φ^{HF}_I sämtliche durch Fehler im Hauptspeicher verursachten Seiteneinlagerungen enthalten. Das sind die fehlerhaften Seiten selbst, sowie die zur Datenintegrität zusätzlich einzulagernden Seiten. Die Rate der durch Fehler beim Paging verursachten Seitenwechsel ist durch Φ^{PF}_I gegeben.

Weitere Iterationsschritte

Nachdem im Fehlermodell mit den verschiedenen Φ_I und $B^R{}_I$ die durch Fehler induzierte Last berechnet wurde, müssen im nachfolgenden Iterationsschritt mit diesen Werten die Gleichungen (7.1-1) bis (7.1-4) neu gelöst werden. Wir erhalten dabei die in Tabelle 8.5 angegebenen Werte:

Tabelle 8.5: Spezifikation des Modells im zweiten Iterationsschritt

Phase 1:	$B_1 = 6.34$	$\mu^{CPU}{}_1 = 174.52$	e_2=0.072	e_3=0.428
Phase 2:	$B_2 = 1.53$	$\mu^{CPU}{}_2 = 138.19$	e_2=0.090	e_3=0.410
Phase 3:	$B_3 = 2.28$	$\mu^{CPU}{}_3 = 122.14$	e_2=0.102	e_3=0.398

Die Ergebnisse sämtlicher Iterationsschritte führen wir in der folgenden Tabelle 8.6 auf. Dabei erkennt man, wie gut die verschiedenen Modellgrößen konvergieren. Als Abbruch-Kriterium des Iterationsverfahrens werden die Differenzen zwischen den in zwei sukzessiven Iterationsschritten berechneten Verweildauern herangezogen.

Tabelle 8.6: Modell-Ergebnisse sämtlicher Iterationsschritte

Iteration		$\bar{V}_r$	$\Phi^{PF}{}_r$	$\Phi^{HF}{}_r$	$\Phi^{PF}{}_r$	$\bar{B}^R{}_r$
1	Phase 1	936.66	0.300	0.795	1.49	0.086
	Phase 2	196.13	0.292	0.502	1.13	0.024
	Phase 3	269.35	0.288	0.664	0.96	0.042
2	Phase 1	959.51	0.302	0.809	1.50	0.088
	Phase 2	200.52	0.293	0.507	1.13	0.025
	Phase 3	275.78	0.288	0.672	0.97	0.042
3	Phase 1	959.84	0.302	0.809	1.50	0.088
	Phase 2	200.58	0.293	0.507	1.14	0.025
	Phase 3	275.89	0.288	0.672	0.98	0.042
4	Phase 1	959.85	0.302	0.820	1.51	0.088
	Phase 2	200.58	0.293	0.510	1.14	0.025
	Phase 3	275.89	0.288	0.675	0.98	0.042

Auch in mit anderen Parametern durchgeführten Analysen brach die Iteration ausnahmslos nach vier bis fünf Schritten ab. Die Berechnung eines einzelnen Iterationsschritts benötigte auf einem Standard IBM-AT (8 MHz, ohne Arithmetikprozessor) ca. 10-15 Sekunden.

8.2 Auswirkungen spezieller Modell-Parameter

In diesem Abschnitt wollen wir genauer untersuchen, wie sich einzelne Modell-Parameter auf Leistungs- und Zuverlässigkeitskenngrößen auswirken. Um übersichtlichere Betrachtungen zu ermöglichen, wollen wir die Ergebnisse der drei Phasen geeignet zusammenfassen. Dazu definieren wir:

$$\bar{V} := \sum_{x} \bar{V}_x \; ; \qquad \bar{B}^R := \sum_{x} \bar{B}^R{}_x \tag{8.2-1a}$$

$$\Phi^F := \sum_{x} \pi_x \cdot \Phi^F{}_x \; ; \; \Phi^{HF} := \sum_{x} \pi_x \cdot \Phi^{HF}{}_x \; ; \; \Phi^{PF} := \sum_{x} \pi_x \cdot \Phi^{PF}{}_x \; . \tag{8.2-1b}$$

In (8.2-1a) werden die auftretenden Zeiten aufsummiert; die Raten in (8.2-1b) werden mit den stationären Phasen-Wahrscheinlichkeiten π_x gewichtet. Bei allen nun folgenden Analyse-Ergebnisse wurden sämtliche Parameter wie in Abschnitt 8.1 angegeben gewählt. Nur die explizit angegebenen Parameter sind variiert worden.

Auswirkung der Fenstergröße

Zunächst wollen wir die bereits in Abschnitt 5.2 prognostizierten Auswirkungen der Fenstergröße betrachten (vgl. auch Abb. 2.5). Einige Größen des Leistungs- und des Zuverlässigkeitsmodells sind für verschiedene Fenstergrößen in Tabelle 8.7 dargestellt:

Tabelle 8.7: Ergebnisse bei verschiedenen Fenstergrößen T

T	$\bar{V}$	Φ^F	Φ^{HF}	Φ^{PF}	$\bar{B}^R$
1	2303.69	0.069	0.075	2.302	0.003
2	2116.45	0.124	0.145	2.112	0.012
3	1941.73	0.167	0.217	1.934	0.025
4	1811.80	0.204	0.299	1.801	0.042
5	1687.63	0.233	0.389	1.662	0.064
6	1583.89	0.257	0.490	1.542	0.090
7	1500.36	0.278	0.604	1.440	0.120
8	1436.32	0.297	0.733	1.338	0.155
10	1391.61	0.336	1.068	1.246	0.248
12	1353.54	0.366	1.463	1.140	0.368

Die in Tabelle 8.7 angegebenen Größen sind in Bild 8.3 dargestellt und gegen die Fenstergröße T aufgetragen In der Abbildung wurde interpoliert (T ist eine natürliche Zahl !). Dabei sind die relative

interpoliert (T ist eine natürliche Zahl !). Dabei sind die relative Abnahme der Verweildauer, sowie die relative Abweichung der Bediendauer dargestellt. Diese berechnen sich nach den folgenden Formeln:

$$\frac{\Delta V}{V_{max}} = \frac{V_{max} - V}{V_{max}} \quad \text{und} \quad \frac{\Delta B}{B_{min}} = \frac{B - B_{min}}{B_{min}} \; . \qquad (8.2\text{-}2)$$

In (8.2-2) ist V_{max} die maximale Verweildauer, die bei Fenstergröße T=1 erreicht wird, und B_{min} bezeichnet die minimale Bedienanforderung, d.h. hier $B_{min} = B^{G}$.

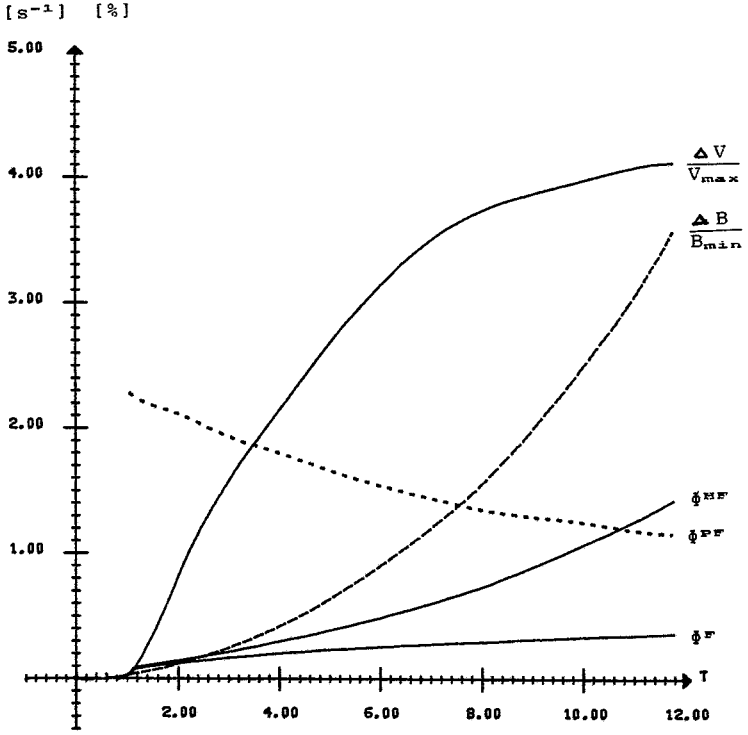


Abb. 8.3: **Änderung der Modell-Parameter bei verschiedenen Fenstergrößen.**

Man erkennt die bereits in Abschnitt 5.2 vorhergesagten Effekte:

- Mit steigender Fenstergröße T sinkt die (nicht aufgetragene) Paging-Rate und damit auch die Verweildauer V eines Auftrags. Sie beträgt bei der Fenstergröße T=10 etwa 40 % weniger als bei T=1, dem Fall, bei dem immer nur eine einzige Seite in den Hauptspeicher eingelagert wird (vgl. Abb.8.3).

- Bei einem großen Fenster T bleibt jede einzelne Seite länger im Hauptspeicher. Deshalb erhöhen sich auch die Raten Φ^F der Hauptspeicherfehler, sowie die Rate Φ^{HF} der durch sie induzierten Seiteneinlagerungen. Weil jeder entdeckte Hauptspeicherfehler meist mehrere Seiteneinlagerungen erfordert, steigt Φ^{HF} in stärkerem Maße als Φ^F.

- Weil die Paging-Rate mit steigendem T abnimmt, verringert sich auch die Rate der Paging-Fehler Φ^{PF}.

- Weil die Einlagerungsdauer D^F_i mit T steigt, nimmt auch die bei einer Fehlerentdeckung zu wiederholende Bediendauer B^R entsprechend zu, (bei T=10 etwa um 2.5 %).

Auswirkung ereignisunabhängiger Fehler

Untersucht man den Einfluß der Rate δ_S der ereignisunabhängigen Fehler auf die Leistungs- und Zuverlässigkeitsindizes, so ergeben sich die in Tabelle 8.8 aufgeführten Ergebnisse. Graphisch sind sie in Abbildung 8.4 dargestellt.

Tabelle 8.8: Ergebnisse bei verschiedenen Fehler-Raten δ_S

δ_S	V	Φ^F	Φ^{HF}	Φ^{PF}	B^R
0.000	1420.99	0.126	0.224	1.323	0.069
0.001	1428.37	0.211	0.446	1.329	0.112
0.002	1436.32	0.297	0.733	1.338	0.155
0.003	1444.86	0.385	1.085	1.348	0.200
0.004	1454.02	0.474	1.503	1.376	0.245
0.006	1474.31	0.658	2.541	1.417	0.340
0.008	1497.51	0.848	3.855	1.470	0.441
0.010	1524.05	1.047	5.459	1.588	0.548

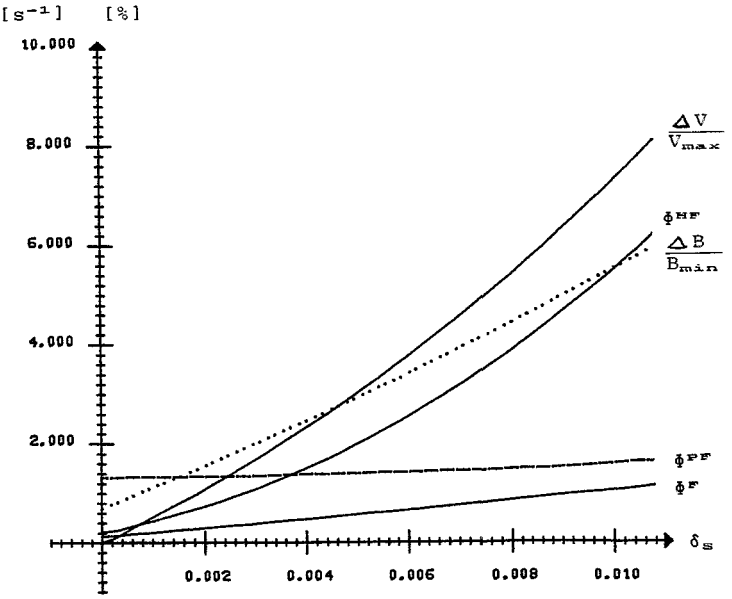


Abb. 8.4: **Ergebnisse des Iterationsverfahrens bei verschiedenen Fehler-Raten δ_S**

In Abbildung 8.4 wird die relative Verweildauer auf die Verweildauer V_{min} bezogen, d.h. die Verweildauer, die bei der Fehler-Rate δ_S=0 erreicht wird. Die Ergebnisse lassen sich wie folgt interpretieren:

- Mit zunehmender Rate δ_S steigt auch die Rate Φ^F für das Auftreten von Hauptspeicherfehlern. Wie oben argumentiert, wächst damit die Rate Φ^{HF} aller zusätzlichen Seiteneinlagerungen exponentiell.

- Die Rate der Paging-Fehler Φ^{PF} erhöht sich nur sehr wenig aufgrund der geringen Zunahme von Φ^{HF}.

- Fast parallel mit dem Ansteigen der Rate Φ^{HF} erhöht sich auch die relative Abweichung der Bediendauer B .

- Ebenso wächst mit δ_S die relative Zunahme der Verweildauer V.

Auswirkung zugriffsinduzierter Fehler

Verändern wir die Wahrscheinlichkeit p_z eines zugriffsinduzierten Fehlers, erhalten wir die in Tabelle 8.9 und Abbildung 8.5 dargestellten Ergebnisse.

Tabelle 8.9: **Ergebnisse bei verschiedenen Wahrscheinlichkeiten für einen zugriffsinduzierten Fehler p_z**

p_z	V	Φ^F	Φ^{HF}	Φ^{PF}	B^R
0.000	1424.03	0.171	0.318	1.325	0.085
0.001	1436.32	0.297	0.733	1.338	0.155
0.002	1449.82	0.423	1.300	1.355	0.227
0.003	1464.47	0.549	2.003	1.396	0.300
0.004	1480.22	0.675	2.826	1.429	0.374
0.006	1514.90	0.927	4.785	1.507	0.526
0.008	1533.71	1.179	7.094	1.599	0.683
0.010	1596.58	1.431	9.691	1.703	0.845

Das Modell dieser Arbeit berücksichtigt Fehler anhand ihrer Auswirkungen auf die Last. Bei der Entdeckung eines Hauptspeicherfehlers kann im allgemeinen nicht unterschieden werden, durch welche Störungs-Art er verursacht wurde. Deshalb wird nicht bei der Behandlung zugriffsinduzierter und ereignisunabhängiger Fehler unterschieden. Somit ergeben sich zwischen Abbildung 8.4 und 8.5 keine signifikanten Unterschiede und die dort gemachten Erklärungen zum Kurvenverlauf gelten auch hier. Allerdings steigen bei unserem Beispiel die Kurven in Abbildung 8.5 stärker, so daß z.B. eine Verdoppelung von p_z stärker durchschlägt als eine von δ_S.

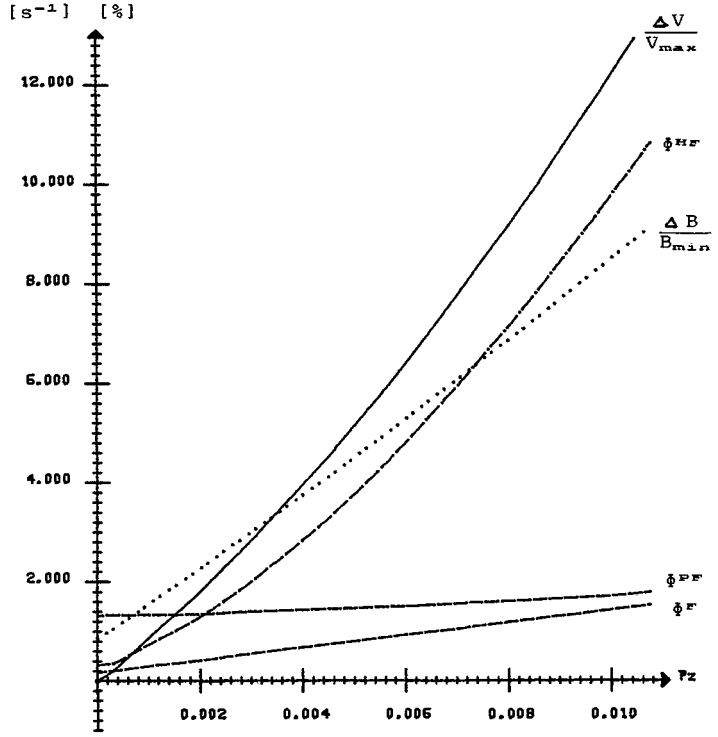


Abb. 8.5: **Ergebnisse des Iterationsverfahrens für verschiedene Wahrscheinlichkeiten p_z eines zugriffsinduzierten Fehlers.**

Auswirkung von Paging-Fehlern

Einen Sonderfall der ereignisabhängigen Fehler bilden die Paging-Fehler. Variiert man die Wahrscheinlichkeit p_F, mit der beim Seitenwechsel ein Fehler induziert wird, ergeben sich die in Tabelle 8.10 und Abbildung 8.6 dargestellten Ergebnisse.

Tabelle 8.10: **Ergebnisse bei verschiedenen Wahrscheinlichkeiten für einen Paging-Fehler p_P**

W-keit p_P	V	Φ^F	Φ^{HF}	Φ^{PF}	B^R
0.00	1427.79	0.296	0.690	0.000	0.155
0.01	1436.32	0.297	0.733	1.338	0.155
0.02	1445.29	0.298	0.781	2.705	0.156
0.03	1454.71	0.299	0.833	4.104	0.156
0.04	1464.59	0.301	0.892	5.570	0.157
0.06	1485.86	0.303	1.031	8.500	0.158
0.10	1535.15	0.309	1.429	14.902	0.161

Die folgenden Wirkungen sind mit einem Ansteigen von p_P verbunden:

- Etwas stärker als linear wächst mit p_P die Rate der auftretenen Paging-Fehler, was mit Gleichung (5.5-7) korrespondiert. Die durch Paging-Fehler induzierten zusätzlichen Seitenwechsel bewirken eine entsprechende Zunahme der Verweildauer eines Auftrags.

- Die anderen betrachteten Größen ändern sich nicht signifikant. Die Bediendauer bleibt praktisch konstant, weil zur Behandlung von Paging-Fehlern keine (Teil-) Wiederholung des Auftrags erforderlich ist. Ebenso wirkt sich die relativ geringe Erhöhung der Verweildauern (<8 %) kaum auf die Fehlerrate Φ^F und damit auch nicht auf Φ^{FH} aus.

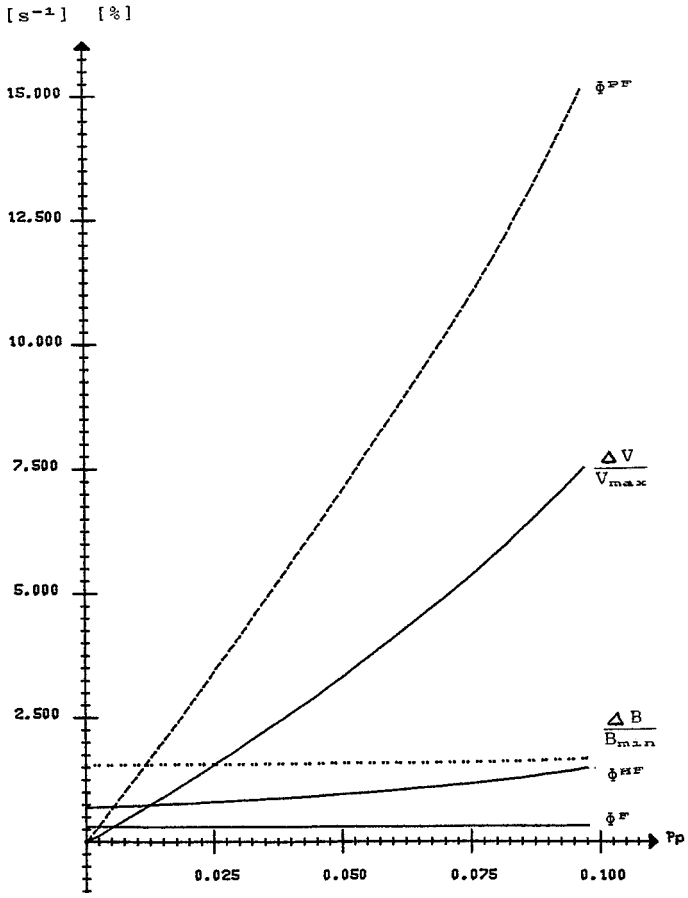


Abb. 8.6: **Ergebnisse des Iterationsverfahrens bei unterschiedlichen Wahrscheinlichkeiten p_P eines Paging-Fehlers**

Auswirkungen auf einen einzelnen Auftrag

Abschließend wollen wir noch in Anlehnung an Abschnitt 7.3 die Auswirkungen betrachten, die ein einzelner Auftrag erleidet, wenn ein Speicherfehler entdeckt wird. Zunächst behandeln wir wieder den Fall der **Fehlerentdeckung beim Auslagern**. In Abhängigkeit von der Zugriffswahrscheinlichkeit auf eine Seite aus Phase 1 ist in Abbildung 8.7 die Verlängerung der Bediendauer, die erhöhte Paging-Rate (zur Wiederherstellung der Datenintegrität, vgl. (7.3-9)) und die für die Teilwiederholung des Auftrags benötigte Verweildauer V^R dargestellt. Die virtuellen Seite der Phase 1 besitzen Zugriffswahrscheinlichkeiten im Intervall [0.02,0.10] (Zwischenwerte wurden interpoliert). In diesem kleinen Bereich ist erwartungsgemäß ein in etwa lineares Ansteigen der zu wiederholenden Bediendauer B^R_i, sowie der resultierenden Verweildauer zu beobachten. Die Paging-Rate nimmt mit B^R_i linear ab.

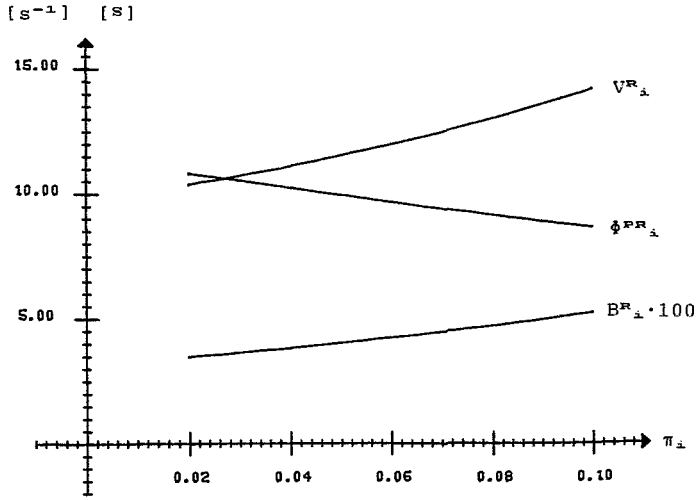


Abb. 8.7: Auswirkungen der Fehlerentdeckung beim Auslagern bei einem einzelnen Auftrag

Der zur Fehlerbehandlung erforderliche Aufwand hängt davon ab, wie lange sich eine betroffene Seite im Hauptspeicher aufgehalten hat. Um ein größeres Spektrum als im vorhergehenden Beispiel zu erhalten, betrachten wir im folgenden eine **Fehlerentdeckung durch Konsistenz-**

überprüfung. Der Parameter in Abbildung 8.8 ist dabei die Anzahl der in B_1 durchgeführten Speicherüberprüfungen.

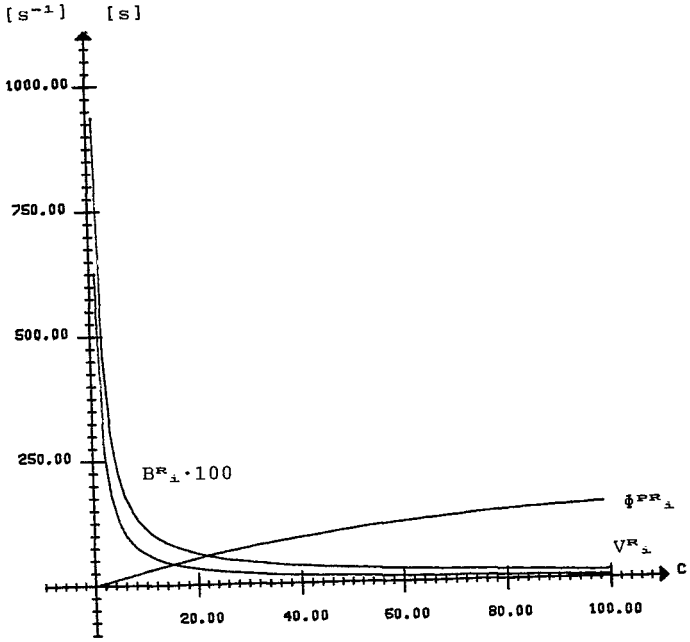


Abb. 8.8: Auswirkungen einer Fehlerentdeckung durch Konsistenzüberprüfung bei einem einzelnen Auftrag

Wird nur eine einzige Speicherüberprüfung und zwar am Ende des Auftrags durchgeführt, so muß der gesamte Auftrag wiederholt werden. Bei sehr vielen Tests ist ein entsprechend geringes roll back erforderlich. Man erkennt, daß gerade mit der Anzahl C der Tests die Wirkung eines einzelnen Fehlers gemäß $B^R=B/C$ abnimmt.

9 Zusammenfassung und Ausblick

In dieser Arbeit haben wir, ein Modell entwickelt, daß die Auswirkungen lastinduzierter Speicherfehler abbildet. Besondere Berücksichtigung fand dabei das Konzept der dynamischen Speicherverwaltung, bei der die Arbeitsmenge eines Auftrags zeitabhängig ist.

Um der Komplexität dieser Aufgabe gerecht zu werden, wurde zunächst eine Modellzerlegung vorgenommen. Weitgehend getrennt voneinander, spezifizierten wir ein Leistungsmodell, ein Modell zum Auftreten von Störungen, und ein Modell zum Entdecken und Behandeln von Fehlern.

Als erstes wurde mit Hilfe von Wahrscheinlichkeits- und Erneuerungstheorie ein Modell zum Auftreten von Störungen im Speicher entwickelt. Dabei wurde besonderer Wert darauf gelegt, möglichst viele für Störungen verantwortliche Größen ins Modell zu integrieren:

- Störungen im Speicher können in Abhängigkeit von bestimmten Ereignissen (wie Seitenzugriffe, oder Seitenwechsel) auftreten.

- Sie können aber auch als ereignisunabhängig angenommen werden, d.h. Speicherseiten sind während ihrer Verweildauer im Hauptspeicher einem durch einen Erneuerungsprozeß modellierbaren Störungsprozeß ausgesetzt.

- Die obengenannten Einflußgrößen sind dabei von bestimmten Parametern des Betriebssystems abhängig, wie z.B. von der Fenstergröße und der vom Scheduler aktivierten Arbeitslast.

- Eine weitere Rolle spielt die Architektur des Speichers, d.h. hier die Organisationsform der Speicher-Chips. Durch sie werden die Auswirkungen einer Störung mitbestimmt.

Die Rate, mit der Fehler entdeckt werden, und der zur Fehlerbehandlung erforderliche Aufwand hängen vornehmlich vom Zeitpunkt ihrer Entdeckung ab. Anhand von drei verschiedenen Modellen untersuchten wir, mit welcher Wahrscheinlichkeit Fehler entdeckt werden. Um die Wirkung von Fehlern auf die Systemleistung bemessen zu können, wurde die zu ihrer Behandlung induzierte Last ermittelt.

Im nächsten Schritt wurde ein einfaches Warteschlangen-Modell entwickelt, um die für unser Fehlermodell erforderlichen Parameter zu errechnen, sowie die Auswirkung von Fehlern auf das Leistungsverhalten quantifizieren zu können. Im zugehörigen Lastmodell berücksichtigten wir insbesondere, daß Aufträgen bzgl. ihres Speicherverhaltens verschiedene Phasen durchlaufen.

Den Wechselwirkungen zwischen Leistungs- und Fehlermodell wurde durch ein iteratives Lösungsverfahren Rechnung getragen.

Insgesamt entwickelten wir ein relativ mächtiges Konzept, mit dem die Häufigkeit von Speicherfehlern und deren Auswirkungen in Abhängigkeit einer Vielzahl von Parametern untersucht werden kann. Insgesamt kann dadurch das Auftreten und die Behandlung von Speicherfehlern sehr differenziert modelliert werden.

Für zukünftige Arbeit scheint interessant, mit diesem Werkzeug nun spezielle Fragestellungen zu untersuchen:

Die Ergebnisse unserer Beispiele aus Kapitel 8 gelten nur für ein ganz spezielles Last- und Systemmodell. Die Auswirkungen weiterere Modell-Parameter (z.B. Bedienraten, Seitenzugriffswahrscheinlichkeiten, usw.) können noch genauer untersucht werden.
Besonders wichtig erscheint in diesem Zusammenhang auch die Frage, wie häufig bei gegebenen Fehler-Raten getestet werden muß, um möglichst hohe Systemleistung und Zuverlässigkeit zu erhalten. Dabei sollte auch der durch Speichertests verursachte Overhead in das Modell einbezogen werden. Ein anderer, interessanter Problemkreis ist die genauere Untersuchung verteilter, z.B. votierender Systeme.

In dieser Arbeit wurde speziell am Beispiel der Speicherfehler gezeigt, wie in differenzierter Weise die Wechselwirkungen zwischen Last und Fehlern modelliert werden können. Für die Modellbildung war dabei eine Zerlegung in Teilmodelle maßgebend. Dabei war es für die Modellierung besonders wirksam, daß wir für jedes Teilmodell eine andere adäquate Theorie zugrunde legen konnten. So wurden zur Lösung der verschiedenen Teilmodelle die

- **Erneuerungstheorie** (für die Modellierung ereignisunabhängiger Störungen),
- **elementare Wahrscheinlichkeitsrechnung** (für die Modellierung ereignisabhängiger Störungen),
- **Markovmodelle** (für ein mehrphasiges IRM-Modell),
- **Warteschlangentheorie** (für das Leistungsmodell)

genutzt.

Diese hybride Modellierung ermöglichte es erst, komplexere Abhängigkeiten zwischen Leistungs- und Zuverlässigkeitsgrößen zu beschreiben; sie erscheint uns auch für andere Performability-Modelle richtungsweisend.

10 Literatur

/AHDU 71/ Aho, A.V., Denning, P.J., Ullmann, J.D.: Principles of optimal page replacements, J. ACM, 18, pp. 80-93, 1971.

/AICH 84/ Aichelmann, F.J.: Fault-Tolerant Techniques for Semiconductor Memory Applications, IBM J. Res. Develop., Vol 28, No.2, pp. 178-183, 1984.

/AVIZ 75/ Avizienis, A.: Architecture of Fault-Tolerant Computers, Proc. of the 5th Intern. Symposium on Fault Tolerant Computing Systems, pp. 3-16, 1975.

/BAEH 88/ Bähring, H.: Zur Fehlerüberdeckung von Prüfsummenverfahren, IT 30(4), pp. 291-299, 1988.

/BARD 77/ Bard, Y.: The modeling of some scheduling strategies of an interactive computer system, Computer Performance, V.M. Chandy, M. Reiser (eds.), North Holland Publishing Company, pp. 113-135, 1977.

/BCMP 75/ Baskett, F., Chandy, K.M., Muntz, R.R., Palacios, F.G.: Open, closed and mixed networks of queues with different classes of customers, J. ACM 22(2), pp. 248-260, 1975.

/BEAU 78/ Beaudry, M.D.: Performance-related reliability measures for computing systems, IEEE Trans. on Comput., Vol. C-27, pp. 540-547, 1978.

/BEIL 88/ Beilner, H.: Leistungsanalyse von Rechensystemen, Kurs 1823, FernUniversität Hagen, 1988.

/BEKU 69/ Belady, L.A., Kuehner, C.J.: Dynamic space sharing in computer systems, CACM (12), pp. 282-288, 1969.

/BLGM 88/ Blaum, M., Goodman, R., McEliege, R.: The Reliability of Single-Error Protected Computer Memories, IEEE Trans. on Comp., Vol. 37, No. 1, pp. 114-119, 1988.

/BOAK 82/ Bolch, G., Akyildiz, I.F.: Analyse von Rechensystemen, Teubner, Stuttgart, 1982.

/BOCC 78/ Bossen, D.C., Chang, L., Chen, C.: Measurement and Generation of Error Correcting Codes for Package Failures, IEEE Trans. Comput., Vol. C-27, no.3, pp. 201-204, 1978.

/BOHS 80/ Bossen, D.C., Hsiao, M.Y.: A System Solution to the Memory Soft Error Problem, IBM J. Res. Develop., Vol. 24, pp. 390-397, 1980.

/BOME 86/ Bose, B., Metzner, J.: Coding Theory for Fault-Tolerant Systems, in /PRAD 86/, Vol. I, pp. 265-336, 1986.

/BOTR 86/ Bobbio, A., Trivedi, K.S.: An Aggregation Technique for the Transient Analysis of Stiff Markov Chains, IEEE Trans. on Comput., Vol. C-35, No. 9, pp. 803-814, 1980.

/BRAN 74/ Brandwajn, A.: A model of a time-sharing virtual memory system solved using equivalence and decomposition methods, Acta Informatica, (4), pp. 11-47, 1974.

/BRAN 85/ Brandwajn, A.: Equivalence and Decomposition in Queueing Systems - A Unified Approach, Performance Evaluation (5), pp. 175-186, 1985.

/BRBC 77/ Brown, R.M., Browne, J.C., Chandy, K.M.: Memory Management and Response Time, CACM 20(3), pp. 153-165, 1977.

/BRIN 77/ Brinch Hansen, P.: Betriebssysteme, Hanser Verlag, München - Wien, 1977.

/BUZE 73/ Buzen, J.P.: Computational algorithms for closed queueing networks with exponential servers, CACM 16, pp. 527-531, 1973.

/CASI 80/ Castillo, X, Siewiorek, D.: A performance reliability model for computing systems, Proc. of the 10th Intern. Symposium on Fault Tolerant Computing Systems, pp. 187-192, 1982.

/CASI 81/ Castillo, X, Siewiorek, D.: Workload, performance, and reliability of digital computing systems, Proc. of the 11th Intern. Symposium on Fault Tolerant Computing Systems, pp.84-89, 1981.

/CASI 82/ Castillo, X, Siewiorek, D.: A workload dependent software reliability model, Proc. of the 12th Intern. Symposium on Fault Tolerant Computing Systems, pp. 279-85, 1982.

/CHHS 84/ Chen, C.L., Hsiao, M.Y.: Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review, IBM J. Res. Develop., Vol 28, No.2, pp. 124- 133, 1984.

/CHOW 83/ Chow, W.M.: Approximation for Large Scale Closed Queueing Networks, Performance Evaluation 3, pp. 1-12, 1983.

/CHRU 84/ Chen, C.L., Rudlege, R.A.: Fault-Tolerant Memory Simulator, IBM J. Res. Develop., Vol 28, No.2, pp. 184-195, 1984.

/CLSK 87/ Cha, S., Leveson, N., Shimeall, T., Knight, J: An Empirical Study of Software Error Detection using Self-Checks, Proc. of the 17th Intern. Symposium on Fault Tolerant Computing Systems, pp. 156-161, 1987.

/COUR 77/ Courtois, P.J.: Decomposability, queueing and computer applications, Academic Press, N.Y. 1977.

/CODE 73/ Coffman, E.G., Denning, P.J.: Operating System Theorie, Prentice Hall, Inc., Englewood Cliffs, NJ, 1973.

/COVA 76/ Courtois, P.J., Vantilborgh, H.: A decomposable model of program paging behaviour, Acta Informatica, 6, pp. 251-275, 1976.

/COX 62/ Cox, D.R.: Renewal Theory, Methuen, London, 1962.

/DALC 79/ Dal Cin, M.: Fehlertolerante Systeme, Teubner, Stuttgart, 1979,

/DALC 82/ Dal Cin, M.: Zuverlässigkeitsanalyse fehlertoleranter Rechnersysteme an Hand von Warteschlangennetzwerk-Modellen, Elektronische Rechenanlagen, pp. 61-67, 1982.

/DEKA 75/ Denning, P., Kahn, K.: A study of locality and lifetime functions, ACM Operating System Review, pp. 207-216, 1975.

/DENN 68/ Denning, P. J.,: The working set model for program behavior, CACM 11, pp. 323-333, 1968.

/DESC 72/ Denning, P.J., Schwartz, S.C.: Properties of the working set model, CACM 15, pp. 191-198, 1972.

/DIGE 87/ Dishon, Y., Georgiou, C.J.: A Highly Available Storage System Using the Checksum Method, Proc. of the 17th Intern.Symposium on Fault Tolerant Computing Systems, pp. 176-181, 1987.

/DOET 81/ Doetsch, G.: Anleitung zum praktischen Gebrauch der Laplace-Transformation und der Z-Transformation, Oldenbourg Verlag, München Wien, 1981.

/DUNK 89/ Dunkel, J.: Betriebssystem-Architektur bei Mikroprozessoren, Kurseinheit 4, Kurs 1703: Mikrorechner-Systeme, FernUniversität Hagen, 1989.

/ELSI 80/ Elkind, S.A., Siewiorek, D.P.: Reliability and Performance of Error-Correcting Memory and Register Arrays, IEEE Trans. on Comput., Vol. C-27, No. 10, pp. 920-927, 1980.

/FELL 68/ Feller, W.: An Introduction to Probability Theorie and its Applications, Vol. I, 3rd ed., Wiley, New York, 1968.

/FERR 78/ Ferrari, D.: Computer System Performance Evaluation, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1978.

/GAED 77/ Gaede, K.-W.: Zuverlässigkeit: Mathematische Modelle, C.Hanser, München - Wien, 1977.

/GAKE 79/ Gay, F.A., Ketelson, M.L.: Performance Evaluation of Graceful Degrading Systms, Proc. of the 9th Intern. Symposium on Fault Tolerant Computing Systems, pp. 51-57, 1979.

/GASC 84/ Gaede, K.-W., Schneeweiß, W.: Erneuerungsprozesse, VDI-Richtlinien 4008, Blatt 8, März 1984.

/GEIL 79/ Geilhufe, M.: Soft errors in semiconductor memories, Digest of papers, COMPCON Spring 79, pp. 210-216, 1979.

/GEMI 80/ Gelenbe, E., Mitrani, I.: Analysis and Synthesis of Computer Systems, Academic Press, London 1980.

/HEMO 81/ HEMOS-Projektgruppe: Heterogene Modellierung von Rechenssystemen, Projektgruppen-Bericht, Universität Dortmund, 1981.

/HEID 83/ Heidtmann, K.: Optimale Testintervalle, Elektronische Rechenanlagen, pp. 205-210, 1983.

/HSIT 87/ Hsueh M.C., Iyer R.K., Trivedi K.S.: A measurement-based performability model for a multiprocessor system, Proc. of 2nd. International Workshop on Applied Mathematics and Performance/ Reliability Models of Computer/Communication Systems, Rome 1987.

/HUMA 87/ Hunt, D., Marinos, P.: a General Purpose Aided Rollback Error Recovery (CARER) Technique, Proc. of the 17th Intern. Symposium on Fault Tolerant Computing Systems, pp. 170-175, 1987.

/HUNG 82/ Hunger, A.: Neues Verfahren zum Selbsttest von Mikroprozessoren, TÜV Rheinland, 1982.

/HUSL 81/ Huslende, R.: A combined evaluation of performance and reliability for degradable systems, ACM/ SIGMETRICS Conf. on Measurement and Modelling of Computer Systems, Las Vegas, pp. 157-164, 1981.

/IYBM 82/ Iyer, R., Butner, S., McCluskey, E.: A statistical failure/load relationship: results of a multicomputer study. IEEE Trans on Comp., Vol. C-31, No.7, pp. 697- 706, July 1982.

/IYRH 86/ Iyer, R., Rosetti, D.J., Hsueh, M.C.: Measurement on modeling of computer reliability as effected by systemactivity, ACM Trans. on Computer Systems, Vol. 4, no. 3, pp. 214-237, Aug. 1986.

/KHMA 86/ Khelalfa, H.M., Mayrhauser, A.: Degradable computer Systems: Performance and Reliability Trade-Offs during Design, NTG Tagung Architektur und Betrieb von Rechensystemen, Stuttgart, März 1986.

/KING 71/ King, W.: Analysis of demand paging algorithms, Proc. IFIP Congr., North-Holland Publ. pp. 485-490, 1971.

/LAPR 85/ Laprie, J.C.: Dependable Computing and Fault Tolerance: Concepts and Terminology, Proc. of the 15th Intern. Symposium on Fault Tolerant Computing Systems, pp. 2-11, 1985.

/LAVE 83/ Lavenberg, S.:Computer Performance Modeling Handbook, Academic Press, New York - London, 1983.

/LAZO 84/ Lazowska, E.D., Zahorjan, J., Graham, G.S., Sevcik, K,C.: Quantitative System Performance, Computer System Analysis using Queueing Network Models, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

/LIHA 84/ Libson, M. R.:, Harvey, H.E.: A General-Purpose Memory Reliability Simulator, IBM J. Res. Develop., Vol 28, No.2, pp. 196-205, 1984.

/MAWO 79/ May, T.C., Woods, M.H.: Alpha-Particle-Induced Soft Errors in Dynamic Memories, IEEE Trans. Electron. Devices, ED-26, No. 4, pp. 10-15, 1976.

/MEWE 88/ Meyer, J.F., Wei, L.: Analysis of Workload Influence on Dependability, Proc. of the 18th Intern. Symposium on Fault Tolerant Computing Systems, pp. 90-96, 1988.

/MEYE 80/ Meyer, J.F.: On Evaluating the Performability of Degradable Computing Systems, IEEE Trans on Comp., Vol. C-29, No.8, pp. 720-731, 1980.

/MEYE 82/ Meyer, J.F.: Closed-form Solutions of Performability, IEEE Trans on Comp., Vol. C-31, No.6, pp. 648-657, 1982.

/MIRA 81/ Miranker, W.L.: Numerical Methods for Stiff Equations, Reidel, 1981.

/MUEL 84/ Mueller-Clostermann, B.: A Decomposition Approach for the Stationary Analysis of Fault Tolerant Queueing Systems, International Workshop on Modeling and Performance Evaluation of Parallel Systems, Grenoble, 1984.

/MUEL 88/ Mueller-Clostermann, B.: An Approximate Product Form for a Class of Degradable Queueing Networks, Performance Evaluation 8, pp. 165-172, 1988.

/MUNA 83/ Munarin, J.A.: Dynamic Workload Model for Performance/Reliability Analysis of Graceful Degrading Systems, Proc. of the 13th Intern. Symposium on Fault Tolerant Computing Systems, pp. 290-295, 1983.

/OHM 79/ Ohm, V.J.: Reliability considerations for semiconductor memories, Digest of papers, COMPCON Spring 79, pp. 207-209, 1979.

/ORRH 70/ Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, San Francisco, London, 1970.

/PRAD 86/ Pradhan, D.K.: Fault-Tolerant Computing, Vol.I, Vol. II, Prentice Hall, Englewood Cliffs, New Jersey, 1986.

/RAND 75/ Randell, B.: System structure for software fault tolerance, IEEE, Trans. on Software Engineering, SE-1, pp.220, 1975.

/REIS 79/ Reiser, M.: Mean value analysis of queueing networks, A new look at an old problem, Proc. 4th Intern. Symp. on Modeling and Performance Evaluation of Computer Systems, Wien, 1979.

/RELA 80/ Reiser, M., Lavenberg, S.S.:Mean value analysis of closed multichain queueing networks, Journal of the ACM (27), pp. 313-322, 1980.

/ROWI 88/ Rohlicek, J.R., Willsky, A.S.: The Reduction of Pertubed Markov Generators: An Algortihm Exposing the Role of Transient States, J. ACM 35(3), pp. 675-696, 1988.

/RUTL 85/ Rutledge, R.A.: Models for the Reliability of Memory with ECC, IEEE ,Proceedings Annual Reliability and Maintainability Symposium 1985.

/SCHO 86/ Schoen, O.: On a Class of Integrated Performance/ Reliability Models based on Queueing Networks, Proc. of the 16th Intern. Symposium on Fault Tolerant Computing Systems, pp. 90-95, 1986.

/SCHN 80/ Schneeweiß, W.: Zuverlässigkeits-Systemtheorie, Köln, Datakontext-Verlag 1980.

/SCHN 81/ Schneeweiß, W.: Binäre Codes mit Redundanz, Kurs 1735, FernUniversität Hagen, 1981.

/SCHN 83/ Schneeweiß, W.: Time Redundancy, Informatik-Bericht Nr.36, FernUniversität Hagen 9/83.

/SCSE 82/ Schneeweiß, W., Seifert, D.: Zuverlässigkeitstheoretische Bewertung von Coderedundanz in fehlertolerierenden Rechenssystemen, GI-Jahrestagung, München, pp.17-31, 1982.

/SCHW 84/ Schweitzer, P.J.: Aggregation Methods for Large Markov Chains, Mathematical Computer Performance and Reliability, G. Iazeolla (ed.), Elsevier Science Publisher B.V., pp. 275-286, 1984.

/SIEW 86/ Siewiorek, D.P.: Architecture of Fault-Tolerant Computers, in /PRAD 86/, Vol. II, pp. 417-467, 1986.

/SISW 82/ Siewiorek, D.P., Swarz, R.S.: The theorie and practice of reliable system design, Digital Press, 1982.

/SKMM 78/ Siewiorek, D.P., Kini, V., Mashburn, H., McConnel, S., Tsao, M.: A Case Studey of C.mmp, Cm*, and C.vmp: Part I - Experiences with Fault Tolerance in Multiprocessor Systems, Proceedings of the IEEE, Vol. 66, No. 10, pp. 1178-1199.

/SPIR 77/ Spirn, J.R.: Program behavior: models and measurements, Elsevier, N. Y. 1977.

/SMTR 88/ Smith, R.M., Trivedi, K.S., Ramesh, A.V.: Performability Analysis: Measures, an Algorithm and a Case Study, IEEE Trans. on Comp., Vol. 37, pp. 406-417, 1988.

/SSLM 84/ de Souza e Silva, E., Lavenberg, S.S., Muntz, R.R.: A Perspective on Iterative Methods for the Approximate Analysis of Closed Queueing Networks, Mathematical Computer Performance and Reliability, G. Iazeolla (ed.), Elsevier Science Publisher B.V., pp. 225-244, 1984.

/STEW 78/ Steward, W.J.: A Comparison of numerical techniques in markov modeling, CACM (21), pp. 144-152, 1978.

/TRGE 83/ Trivedi, K.S., Geist, R.M.: Decomposition in Reliability Analysis of Fault Tolerant Systems, IEEE Trans. on Reliability., Vol. R-32, pp. 463-468, 1983.

/TRIV 82/ Trivedi, K.S.: Probability & Statistics with Reliability, Queuing and Computer Science Applications, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

/TRIV 85/ Trivedi, K.S.: Modeling and Analysis of Fault-Tolerant Systems, in Portier, D. (ed.): Modelling Techniques and Tools for Performance Analysis, North Holland, 1985.

/VANT 74/ Vantilborgh, H.: On the working set size and its normal approximation, BIT(14), pp.240-251, 1974.

/WEIN 68/ Weingarten, A.: The Eschenbach drum scheme, Comm. ACM 9, pp. 509-512, 1968.

/WENS 78/ Wensley, J.H., Lamport, L., Goldberg, J., Grenn, M.W., Levitt, K.N., Melliar-Smith, P.M., Shostak, R.E., Weinstock, C.B.: SIFT - The Design and Analysis of a Fault-Tolerant Computer for Aircraft Control, Proceedings of the IEEE, Vol. 66, pp. 1240, 1978.