KERNFORSCHUNGSANLAGE JÜLICH GmbH

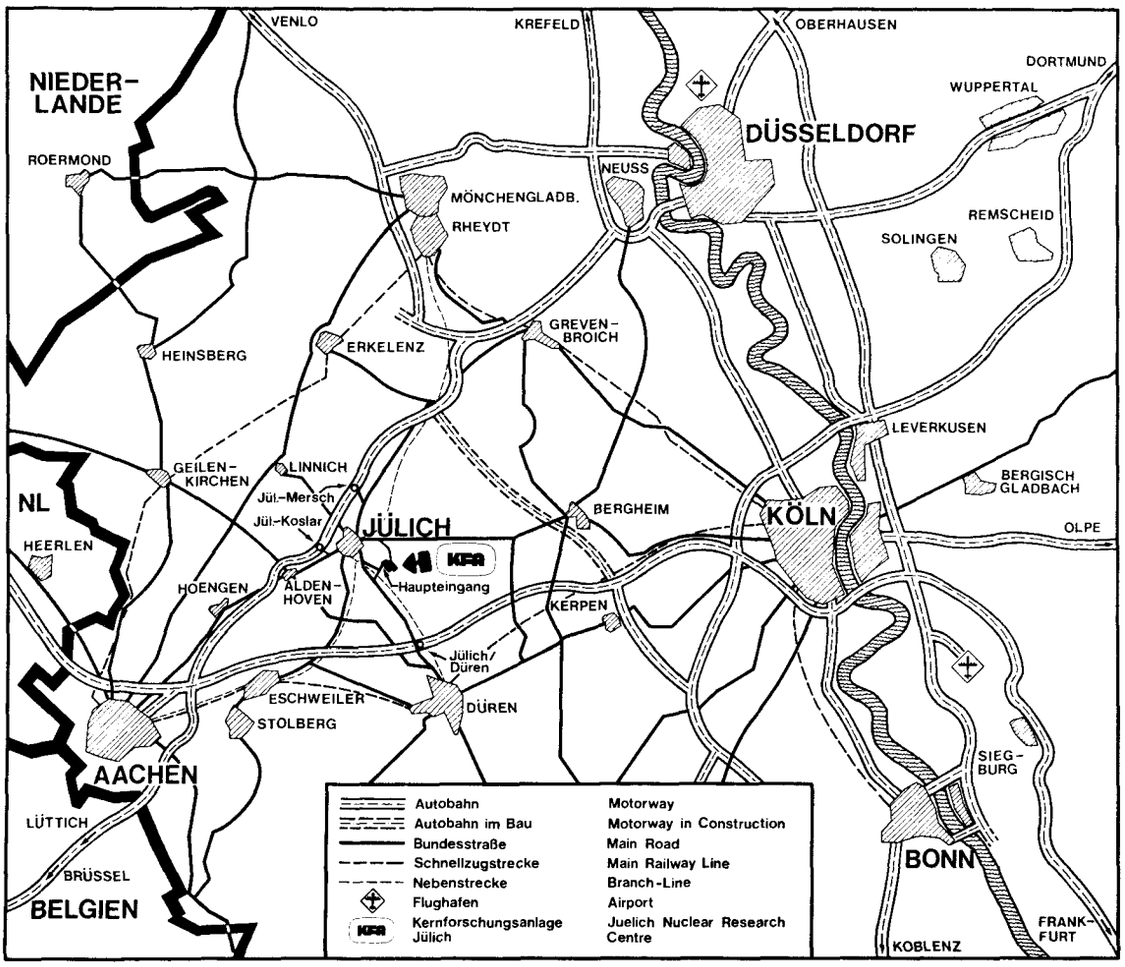
Zentralinstitut für Angewandte Mathematik

Untersuchungen über Seitenaustauschverfahren unter besonderer Berücksichtigung von Prepaging Konzepten und der Bewertung ihrer Leistungsfähigkeit

von

D. Conrads

Jül - 1657
Mai 1980
ISSN 0366-0885



Als Manuskript gedruckt

Berichte der Kernforschungsanlage Jülich - Nr. 1657

Zentralinstitut für Angewandte Mathematik Jül - 1657

Zu beziehen durch: ZENTRALBIBLIOTHEK der Kernforschungsanlage Jülich GmbH

Postfach 1913 · D-5170 Jülich (Bundesrepublik Deutschland)

Telefon: 02461/611 · Telex: 833556 kfa d

Untersuchungen über Seitenaustauschverfahren unter besonderer Berücksichtigung von Prepaging Konzepten und der Bewertung ihrer Leistungsfähigkeit

von

D. Conrads

D 290 (Diss. Uni. Dortmund)

Diese Arbeit entstand am Zentralinstitut für Angewandte Mathematik der Kernforschungsanlage Jülich nach einem von Prof. Richter an der Universität Dortmund veranstalteten Seminar über Arbeitsmengenmodelle.
Danken möchte ich an dieser Stelle besonders Herrn Prof. Richter, der die Thematik dieser Arbeit angeregt und die Dissertation betreut hat, Herrn Prof. Unger, der sich als Berichterstatter zur Verfügung gestellt hat, und Herrn Dr. Hoßfeld, der sie jederzeit gefördert hat.
Dank gebührt auch Herrn Schippers (Universität Dortmund), der sein Programm zur Erstellung von Referenzketten den Erfordernissen dieser Arbeit angepaßt und zur Verfügung gestellt hat, sowie Frl. Steffes und Herrn Erkens für ihre Mitarbeit bei der Programmierung der Simulationsverfahren und Frau Peters, die die Texte mit großer Sorgfalt geschrieben hat.

Kurzfassung

Seitenorientierte virtuelle Speicher sind ein verbreitetes Konzept der Speicherorganisation heutiger Betriebssysteme; in solchen Systemen sind Seitenaustauschverfahren, die den Datenfluß zwischen Hauptspeicher und Sekundärspeicher steuern, wichtige Komponenten. In nahezu allen existierenden Systemen geschieht das Laden einer Seite vom Sekundärspeicher in den Hauptspeicher auf Anforderung (demand); Prepaging kann dazu eine Alternative sein, die bisher jedoch nur wenig Beachtung gefunden hat.

Neben Untersuchungen allgemeiner Eigenschaften (Regularität, Realisierungsaufwand etc.) bildet die relative Performance von Demand Paging Verfahren und Prepaging Verfahren einen Schwerpunkt dieser Arbeit. Mehrere optimale Prepaging Verfahren werden definiert, und es wird nachgewiesen, daß sie nach den zugrundegelegten Performance-Kriterien entsprechenden Demand Paging Verfahren grundsätzlich überlegen sind. In der Klasse der nichtoptimalen (realisierbaren) Seitenaustauschverfahren werden Verfahren vorgestellt, die aus einer Kombination bekannter Ersetzungsstrategien mit einer Ladestrategie entstehen, die auf der Beobachtung des Seitenreferenzverhaltens basiert. In umfangreichen Simulationen haben sich diese Verfahren den zugrundeliegenden Demand Paging Verfahren für weit verbreitete Programmverhaltensweisen überlegen gezeigt.

Abstract

Paged virtual memory is a commonly realized concept of storage organization in existing operating systems; therefore paging algorithms which control the flow of information between memory and secondary storage are major components of these systems. In almost all existing systems page transport from secondary storage to main memory is realized on a demand basis; prepaging is an alternative which has not yet received too much attention.

In addition to investigations of general properties (absence of abnormal behavior, implementation overhead, etc.) the relative performance of demand paging algorithms and prepaging algorithms is focussed in this report. Several optimal prepaging algorithms are defined and proved to outperform comparable optimal demand paging algorithms with regard to the selected performance criteria. In the class of non-optimal (realizable) paging algorithms a set of prepaging algorithms is introduced which consist of a combination of one of the well-known replacement policies with a prepaging fetch policy based on information which can be extracted from the program reference behavior observed during prior execution. In extensive simulations the performance of these algorithms is shown to be superior to that of the basic demand paging algorithms for a wide range of program behavior.

Inhalt

1 EINLEITUNG 1

1.1 Einordung 1

1.2 Zielsetzung und Ergebnisse der Arbeit 4

2 DEMAND PAGING 6

2.1. Performance - Maße 6

2.1.1 Fehlseitenrate 6
2.1.2 Mittlere Gültigkeitsdauer 8
2.1.3 Space-time Produkt 9
2.1.4 Prozessor-Intensität 13

2.2 Demand Paging Verfahren fester Arbeitsspeichergröße 15

2.2.1 Nichtoptimale Verfahren 15
2.2.2 Optimale Verfahren 16

2.3 Demand Paging Verfahren variabler Arbeitsspeichergröße 18

2.3.1 Nichtoptimale Verfahren 19
2.3.2 Optimale Verfahren 28

2.4 Realisierungsaufwand und abgeleitete Verfahren 33

2.5 Regularitätseigenschaften 38

2.5.1 Regularität bei Verfahren fester Arbeitsspeichergröße 38
2.5.2 Regularität bei Verfahren variabler Arbeitsspeichergröße 40
2.5.3 Regularitätseigenschaften der vereinfachten Seitenaustauschalgorithmen 45

2.5.3.1 MWS-Verfahren 45
2.5.3.2 Typ-LRU-Verfahren mittels zeitgesteuerter UIC's 47
2.3.3.3 2-Klassen LRU-Verfahren auf der Basis von Fehlseitenbedingungen 52

2.6 Verfahren zur Simulation 55

2.6.1 LRU-Verfahren 55
2.6.2 OPT-Verfahren 57
2.6.3 DWS-Verfahren 58
2.6.4 VMIN-Verfahren 64
2.6.5 PFF-Verfahren 65

3 PREPAGING 66

3.1 Performance-Maße 67

3.1.1 Space-time Produkt 67
3.1.2 Fehlseitenrate (FSR) 70
3.1.3 Modifizierte Fehlseitenrate (MFSR) 70

3.2 Optimale Prepaging Verfahren 72

3.2.1 Verfahren fester Arbeitsspeichergröße 72

3.2.1.1 Allgemeines 72
3.2.1.2 Das PREOPT-Verfahren 76
3.2.1.3 Das OPR-Verfahren 87

3.2.2 Verfahren variabler Arbeitsspeichergröße 94

3.2.2.1 Allgemeines 94
3.2.2.2 Das VMINPP1-Verfahren 96
3.2.2.3 Das VMINPP2-Verfahren 101

3.3 Nichtoptimale Prepaging Verfahren 106

3.3.1 Motivation - Allgemeines 106
3.3.2 Das Folgeseiten-Verfahren 109

3.3.2.1 Grundsätzliches 109
3.3.2.2 Erstellung und Überprüfung der Verkettungsinformation 112
3.3.2.3 Nutzung der Verkettungsinformation 118
3.3.2.4 Zusatzinformation der Simulation 120
3.3.2.5 Spezielle Verfahren 121
3.3.2.6 Aufwandsbetrachtungen 127

4 MESSUNGEN - ERGEBNISSE 134

4.1 Allgemeines 134

4.2 Vergleichende Messungen bei optimalen Verfahren 139

4.2.1 Verfahren fester Arbeitsspeichergröße 139
4.2.2 Verfahren variabler Arbeitsspeichergröße 141

4.3 Vergleichende Messungen bei nichtoptimalen Verfahren 145

4.3.1 Grundauswertung 145
4.3.2 Einfluß der Programmlänge 160
4.3.3 Speicherung der Verkettungsinformation 163
4.3.4 Einfluß der Seitengröße 166
4.3.5 Alternative Verkettungen 173

4.4 Zusammenfassung 179

5 AUSBLICK 181

LITERATUR 185

ANHANG A Tabellen der Performance-Daten der optimalen Verfahren 192

ANHANG B Bezeichnungen, Definitionen, Begriffserläuterungen 203

ANHANG C Verzeichnis der Tabellen und Bilder 209

1 EINLEITUNG

1.1 Einordnung

Basis der Untersuchungen ist ein System mit seitenverwaltetem virtuellem Speicher. Die Anfangsphase der Entwicklung solcher Systeme war geprägt von der Hoffnung, Kernspeicher sparen zu können (bzw. die negativen Folgen zu geringen Speicherplatzes - wie z. B. intensive Verwendung von Overlays - zu mildern). Die erhebliche Komplexität solcher Systeme, der damit verbundene Overhead (gerade auch bei den frühen Implementationen) und der erhöhte Eigenbedarf an Speicherplatz sowie die bei Überstrapazierung des Prinzips vorhandene schlechte Performance und Neigung zu instabilen Systemzuständen haben gezeigt, daß diese Hoffnung trügerisch war. Spätestens seit etwa 1973 wird denn auch von namhaften Autoren immer wieder darauf hingewiesen, daß Paging kein Ersatz für Kernspeicher ist. Dies wird nachdrücklich unterstrichen durch den Platzbedarf moderner Betriebssysteme mit virtuellem Speicherkonzept (wie z. B. das MVS von IBM), die schon von daher auf Systemen damals üblichen (und möglichen) Speicherausbaues nicht sinnvoll zu implementieren wären. Dieser Aspekt ist aber auch durch die Entwicklung auf dem Hardware -Sektor, sowohl was die Ausbaubarkeit der Systeme als auch die Kosten angeht, stark abgewertet worden, ein Prozeß, der noch nicht beendet ist.

HANSEN /H1/ weist darauf hin, daß von den Entwicklungsstufen der Betriebssysteme (grob vom Einbenutzersystem über Mehrbenutzersysteme mit festen und variablen Speicherbereichen zu Paging Systemen) die unterste Stufe, die die Auslastung der Zentraleinheit ermöglicht, die geeignetste ist, weil dort aufgrund des geringeren Overheads der größte relative Anteil der erbrachten Rechenleistung den Benutzern zukommt. In Forschungseinrichtungen ist es wegen des Vorhandenseins nahezu beliebig rechenintensiver Programme im allgemeinen bereits auf der ersten Multiprogramming gestattenden Stufe der Betriebssysteme möglich gewesen, eine hohe CPU-Auslastung zu erzielen, zumindest in reinen Batch-Systemen bei nicht minimalen Turnaround-

Zeiten.

Unbestritten der Tatsache, daß im Forschungsbereich immer ein leistungsfähiger Stapelbetrieb notwendig sein wird, ist die Vielfalt der geforderten Datenverarbeitungsleistungen gerade in diesem Bereich in den letzten Jahren stark gestiegen (vgl. /H4/), wobei der Gesichtspunkt einer optimalen Systemauslastung oft hinter dem des effizienten Personaleinsatzes zurücksteht. Das Faszinierende am Konzept seitenverwalteter virtueller Speicher - was ihm letztlich auch zum endültigen Durchbruch verholfen haben dürfte - ist, daß damit ein einheitliches Speicherkonzept zur Verfügung steht, daß all den vielfältigen und teilweise divergenten Anforderungen gerecht wird; den gerade in komplexen Vielbenutzersystemen besonders wichtigen Grundsätzen der Wahrung der individuellen Integrität (protection) und der gemeinsamen Nutzung von Programmen und Daten (program and information sharing) werden Paging Systeme in geradezu idealer Weise gerecht.

Wie gut die Performance eines Paging Systems ist, hängt in erster Linie von Programmeigenschaften und dem Seitenaustauschalgorithmus ab, die beide durch eine Reihe von Faktoren bestimmt und beeinflußbar sind. Wenn auch - wie etwa in /G3/ ausgeführt wird - Veränderungen der Programmeigenschaften das größere Verbesserungspotential bieten, so kann aber nicht davon ausgegangen werden, daß die Programmqualität sich - insbesondere bzgl. der für virtuelle Systeme wichtigen Eigenschaft lokalen Verhaltens - in absehbarer Zeit ändert: Programmrestrukturierung, durch die eine Verbesserung erzielt werden kann, ist viel zu aufwendig für eine breite Anwendung, Compiler und Linker haben nur einen verhältnismäßig geringen Einfluß (bei den heute gebräuchlichen Sprachen) und Verbesserungen bei der Programmerstellung würden sich - selbst wenn sie auf breiter Front durchgeführt würden - erst mittelfristig stärker auswirken. So kommt vorerst den betriebssystemseitigen Möglichkeiten eine besondere Bedeutung zu, und hierbei besonders den Seitenaustauschalgorithmen. Von Wichtigkeit ist die Anpassungsfähigkeit an das dynamische Programmverhalten und damit zusammenhängend die Qualität der Schätzung der Programmlokalitäten. Eine

weitere prinzipielle Möglichkeit, die Performance zu verbessern, besteht darin, den erforderlichen Aufwand und Overhead für den Austausch von Programmlokalitäten zwischen den Speicherstufen zu vermindern. Ein Anatz dafür ist Prepaging. Prepaging kann als Versuch einer dynamischen Restrukturierung aufgefaßt werden, denn während bei der Restrukturierung zusammengehörende Programmteile physisch zusammengefaßt werden, werden beim Prepaging Programmteile, die zusammengehören (bei den optimalen Verfahren) oder von denen unterstellt wird, daß sie zusammengehören, logisch zusammengefaßt und bei Transportvorgängen als Einheit aufgefaßt. Daß dem Prepaging grundsätzlich ein Verbesserungspotential innewohnt, zeigen die optimalen Verfahren, die unter gleichen Bedingungen erheblich bessere Ergebnisse erzielen, als die optimalen Demand Paging Verfahren. Wie weit dieses Potential in der Praxis nutzbar gemacht werden kann, hängt entscheidend davon ab, wie zuverlässig Programmlokalitäten im voraus zu schätzen sind.

1.2 Zielsetzung und Ergebnisse der Arbeit

Die vorliegende Arbeit befaßt sich mit Paging Verfahren, wobei vergleichende Untersuchungen von Prepaging Verfahren und Demand Paging Verfahren im Mittelpunkt stehen. Neben Untersuchungen allgemeiner Eigenschaften (Regularitätseigenschaften, Realisierungsaufwand etc.) bilden Untersuchungen der relativen Performance von Verfahren beider Kategorien einen Schwerpunkt. In diesem Zusammenhang werden mehrere neue Prepaging Verfahren definiert, insbesondere mit dem Ziel, Prepaging Verfahren zu finden, die mit geringem Aufwand realisierbar und entsprechenden Demand Paging Verfahren nicht nur nicht unterlegen, sondern sogar überlegen sind.

Voraussetzung für einen aussagekräftigen Vergleich der Performance beider Verfahrenskategorien ist die Benutzung eines Performance-Maßes, das in der Lage ist, Vorteile und Nachteile beider Kategorien angemessen zu erfassen; deshalb werden einige der bekannten Performance-Maße ausführlich analysiert und auf ihre Eignung hin untersucht, die Performance von Prepaging Verfahren zu beschreiben, und es wird schließlich eine 'Modifizierte Fehlseitenrate' als Performance-Maß vorgeschlagen, die bei Anwendung auf Demand Paging Verfahren mit dem bekannten Performance-Maß 'Fehlseitenrate' identisch ist.

Kapitel 2 befaßt sich mit Demand Paging. Hierbei nehmen Regularitätsuntersuchungen einen breiten Raum ein. Die Frage nach der Regularität von Paging Verfahren, die seit der Entdeckung der FIFO-Anomalie durch BELADY /B 6/ ins Bewußtsein gerückt ist und durch die Entdeckung anomalen Verhaltens beim PFF-Verfahren durch FRANKLIN, GRAHAM und GUPTA /F 6/ neue Aktualität erhalten hat, wird vor allem im Hinblick auf vereinfachte Realisierungen der bekannten Ersetzungsstrategien DWS und LRU untersucht, wie sie aus Aufwandsgründen in realen Betriebssystemen zum Einsatz kommen; dabei zeigt es sich, daß durch solche Vereinfachungen, die für die Praxis notwendig sind, häufig die Regularitätseigenschaften der Originalverfahren verlorengehen. Abschließend wird für Demand Paging Verfahren fester und variabler

Arbeitsspeichergröße jeweils ein Simulationsverfahren entwickelt, das es gestattet, eine Vielzahl von Performance-Daten für beliebige Parameterwerte ab einem vorgebbaren Grenzwert mit stark reduziertem Aufwand zu berechnen. Voraussetzung dafür ist, daß der durch das Paging Verfahren definierte Working Set bezüglich des Verfahrensparameters einer Einschlußbedingung genügt; angewendet wird dieses Konzept bei der Simulation des LRU- und des DWS-Verfahrens.

Kapitel 3 ist dem Themenkomplex Prepaging gewidmet. Vom Potential her haben Prepaging Verfahren erhebliche Vorteile gegenüber Demand Paging Verfahren, was beim Vergleich der optimalen Verfahren beider Kategorien eindeutig nachgewiesen wird; praktisch anwendbare Prepaging Verfahren scheitern häufig jedoch an der Schwierigkeit, genügend zuverlässige Informationen über zukünftiges Programmverhalten bereitzustellen.

Es werden zunächst zwei optimale Verfahren fester Arbeitsspeichergröße (PREOPT und OPR) definiert und ihre Optimalität (nach verschiedenen Kriterien) nachgewiesen; des weiteren werden unter der Bezeichnung VMINPP1 und VMINPP2 zwei auf dem VMIN-Verfahren /P 4/ aufbauende optimale Prepaging Verfahren variabler Arbeitsspeichergröße vorgestellt und an allen Verfahren Regularitätsuntersuchungen durchgeführt. Messungen zeigen, daß die qualitativ nachgewiesene Überlegenheit der Prepaging Verfahren quantitativ erheblich ist.

In der Klasse der realisierbaren Verfahren wird eine Menge von 'Folgeseiten-Verfahren' genannten Verfahren vorgestellt. Diese Verfahren beruhen auf einer logischen Verkettung der in Sequenz Fehlseitenbedingungen erzeugenden Seiten, wobei diese Verkettungsinformationen, sobald und in dem Umfange, wie sie vorhanden sind, durch Prepaging genutzt werden. Auf der Basis dieser Vorgehensweise entstehen durch Kombination mit bekannten Ersetzungsstrategien verschiedene Folgeseiten-Verfahren, die sich in umfangreichen Meßreihen durchweg den zugrunde liegenden reinen Demand Paging Verfahren überlegen gezeigt haben.

2 DEMAND PAGING

2.1 Performance-Maße

2.1.1 Fehlseitenrate (page fault rate)

Die Fehlseitenrate FSR ist definiert durch die Anzahl der Fehlseitenbedingungen FS pro virtueller Zeiteinheit. Die Anzahl der Fehlseitenbedingungen ist abhängig vom Programm (charakterisiert durch die Referenzkette ω), vom Seitenaustauschverfahren (A) und von der Größe des zur Verfügung stehenden Arbeitsspeichers (m). Bei Verfahren mit variabler Arbeitsspeichergröße ist die vorgegebene Größe m durch die mittlere Arbeitsspeicherbelegung $\overline{m}$ zu ersetzen, die ihrerseits von einem Verfahrensparameter T abhängig ist.

Sei

$$\Delta(A, m, \omega, t) = \begin{cases} 1, \text{ falls } r_t \notin S_{t-1} \text{ (d.h. eine Fehlseitenbed. produziert)} \\ 0 \text{ sonst.} \end{cases}$$

Dann ist

$$FS(A, m, \omega) = \sum_{t=1}^{|\omega|} \Delta(t) \qquad (2.1.1)$$

und

$$FSR(A, m, \omega) = \frac{1}{|\omega|} FS(A, m, \omega) = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} \Delta(t) . \qquad (2.1.2)$$

Wegen der Annahme $S_0 = \emptyset$ gelten für die Fehlseitenrate die folgenden oberen und unteren Schranken

$$\frac{|N|}{|\omega|} \leq FSR(A, m, \omega) \leq 1 .$$

Sehr verbreitet ist für die Performance-Beschreibung eines Verfahrens die Angabe der Fehlseitenrate über der Arbeitsspeichergröße (bei festem ω). Bei größenvariablen Verfahren wird die Fehlseitenrate über der mittleren Working Set Größe aufgetragen, wobei

$$\bar{m} = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} |S_t(A, \omega)| \qquad (2.1.3)$$

ist.

Die Kurve hat grundsätzlich den in Bild 2.1 gezeigten Verlauf.

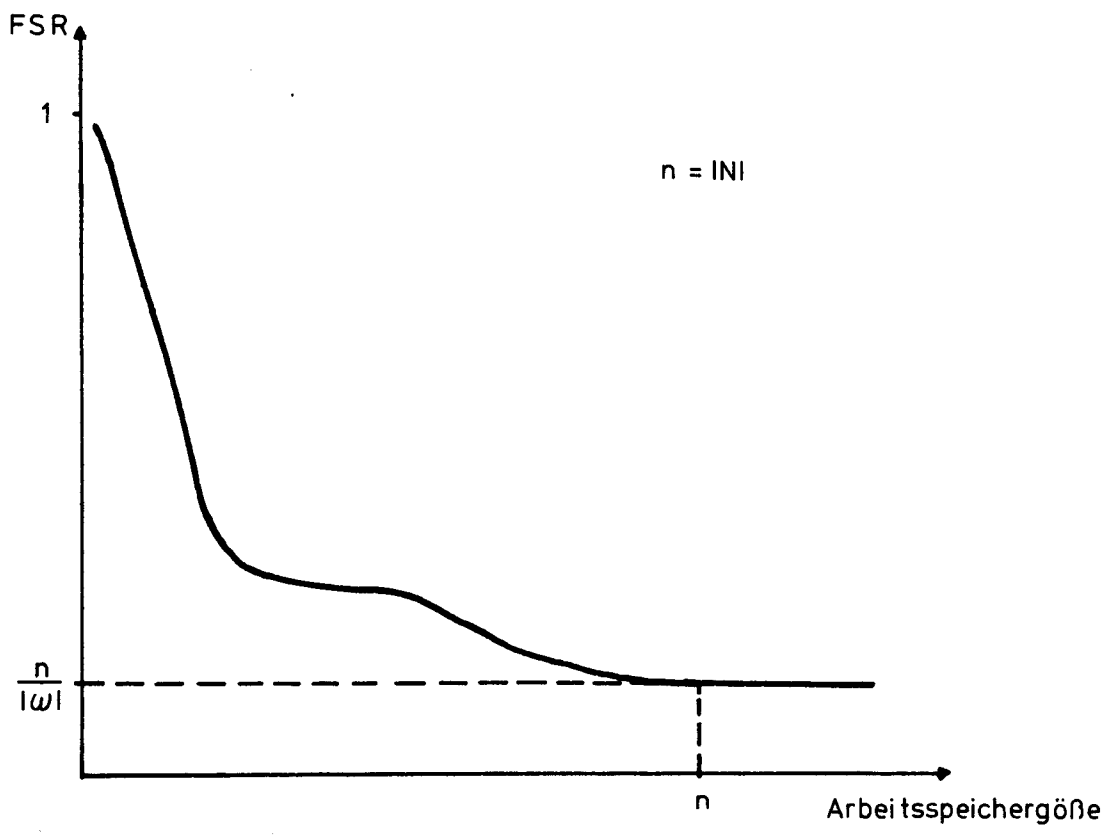


Bild 2.1
Fehlseitenrate als Funktion der Arbeitsspeichergröße

2.1.2 Mittlere Gültigkeitsdauer (lifetime function)

Der Ausführungsprozeß kann auch durch die Folge der zeitlichen Abstände von Fehlseitenbedingungen charakterisiert werden. Aus diesen Werten kann durch Durchschnittsbildung ein mittlerer Abstand gebildet werden, der - bei vorgegebenem Verfahren und Programm - von der Arbeitsspeichergröße abhängig ist.

Die mittlere Gültigkeitsdauer MGD (A, ω, m) ist der mittlere Fehlseitenabstand, der sich ergibt, wenn die Referenzkette ω durch den Seitenaustauschalgorithmus A in einen Arbeitsspeicher der Größe m Seiten verarbeitet wird.

Unter der - in dieser Arbeit grundsätzlich gültigen - Annahme $S_0 = \emptyset$, unter der jede erste Referenz zu einer Fehlseitenbedingung führt, ist die mittlere Gültigkeitsdauer genau die reziproke Funktion der Fehlseitenrate, d.h.

$$MGD(m) = \frac{1}{FSR(m)} \quad . \tag{2.1.4}$$

Man kann erwarten, daß kleine Arbeitsspeicher zu kurz aufeinanderfolgenden Fehlseitenbedingungen und damit auch zu einer kleinen mittleren Gültigkeitsdauer führen werden; andererseits wird - wenn der Arbeitsspeicher genügend groß ist, um alle Programmseiten aufnehmen zu können - nach der Aufbauphase die Gültigkeitsdauer des Speicherzustandes in der Größenordnung der Programmdauer sein, die mittlere Gültigkeitsdauer gleich $\frac{|\omega|}{|N|}$ betragen. Zwischen diesen beiden Extremen würde man einen monoton wachsenden Verlauf der mittleren Gültigkeitsdauer erwarten. In Kapitel 2.5 wird gezeigt, daß dies - bzw. die gleichwertige Aussage eines monoton fallenden Verlaufs der Fehlseitenrate - richtig ist für reguläre Verfahren.
Generell verläuft die mittlere Gültigkeitsdauer in Abhängigkeit der Arbeitsspeichergröße etwa wie in Bild 2.2 dargestellt.

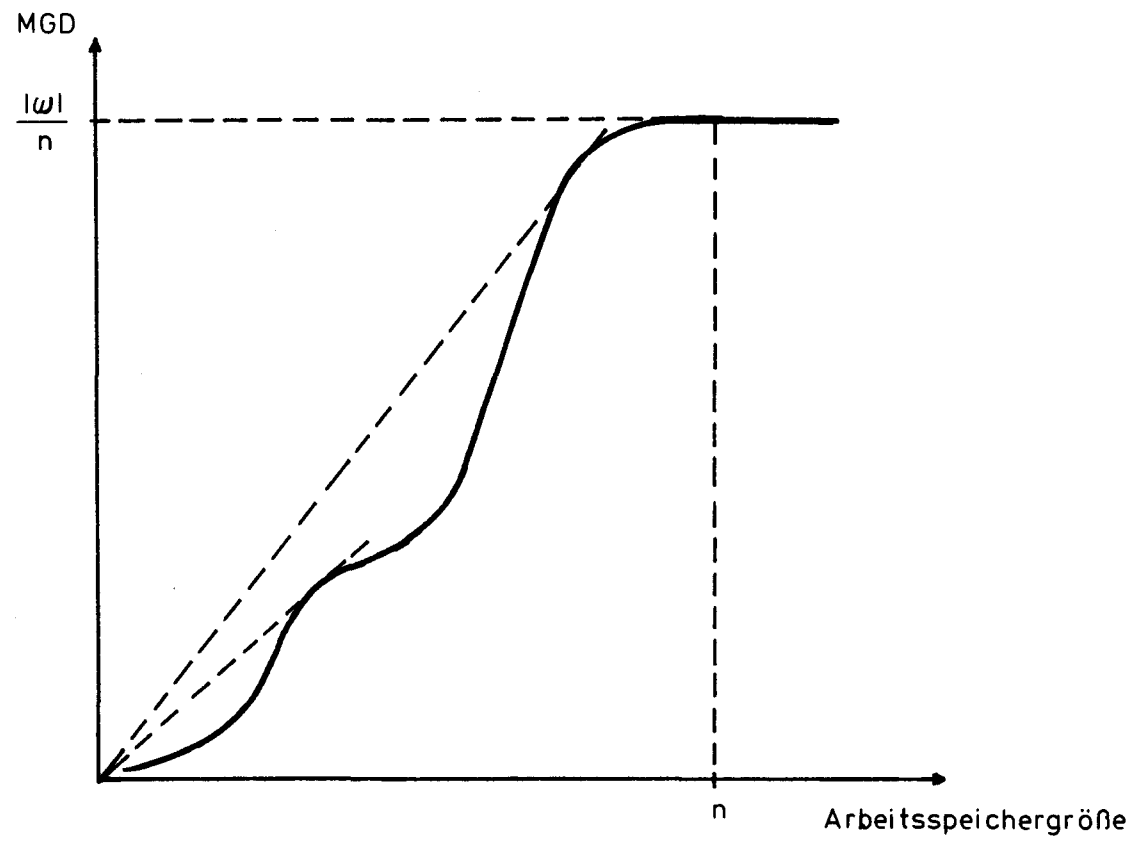


Bild 2.2
Mittlere Gültigkeitsdauer als Funktion der Arbeitsspeichergröße

Eine Reihe von Arbeiten /B5, D4, D5, G3/ beschäftigt sich mit den Zusammenhängen zwischen ausgezeichneten Punkten der Kurve der mittleren Gültigkeitsdauer und Programm- und Verfahrenseigenschaften.

2.1.3 Space-time Produkt

Das Space-time Produkt ist ein Indikator für die Kosten der Arbeitsspeicherbelegung während der Programmausführung. Es ist wie alle Performance-Maße vom Seitenaus-

tauschalgorithmus A, vom Programm (d.h. von der Referenzkette ω) und von der mittleren Größe $\bar{m}$ des belegten Arbeitsspeichers abhängig.

Das Space-time Produkt kann in Prozeßzeit und in Realzeit gemessen werden.

(a) Space-time Produkt in Prozeßzeit

Setzt man der Einfachheit halber die Kosten für die Aufenthaltsdauer einer Seite für eine virtuelle Zeiteinheit im Arbeitsspeicher zu 1, so ist das Space-time Produkt gegeben durch

$$STP_V = STP_V(A, \omega, \bar{m}) = \sum_{t=1}^{|\omega|} |S_t(A, \omega, \bar{m})|. \qquad (2.1.5)$$

Hierbei sind nur solche Zeiten berücksichtigt, während deren sich das Programm in der Ausführung befunden, d.h. Service durch die CPU erhalten hat.

(b) Space-time Produkt in Realzeit

Aussagekräftiger als in virtueller Zeit ist das Space-time Produkt gemessen in Realzeit. Dieses Space-time Produkt ist das einzige der bekannten Performance-Maße, bei dem manifest wird, daß Ladevorgänge unterschiedliche Kosten verursachen, je nachdem, wieviele System-Resourcen (in diesem Falle Seitenrahmen des Arbeitsspeichers) während der Dauer des Ladevorganges belegt sind.

Sei Q die Zeit (gemessen in virtuellen Zeiteinheiten), die notwendig ist, um eine Seite vom Sekundärspeicher in den Arbeitsspeicher zu laden, dann ist

$$\begin{aligned} STP_R(A, \omega, \bar{m}) &= STP_V(A, \omega, \bar{m}) + Q \sum_{t=1}^{|\omega|} \Delta(t) \cdot |S(A, \omega, \bar{m})| \\ &= \sum_{t=1}^{|\omega|} |S_t| + \sum_{t=1}^{|\omega|} Q \cdot \Delta(t) \cdot |S_t| \\ &= \sum_{t=1}^{|\omega|} |S_t| \cdot (1 + Q \cdot \Delta(t)) \qquad (2.1.6) \end{aligned}$$

$$\text{mit } \Delta(t) = \begin{cases} 1, \text{ falls } t \text{ Zeitpunkt einer Fehlseitenbedingung ist} \\ \qquad (\text{d.h. } r_t \notin S_{t-1}) \\ 0 \text{ sonst.} \end{cases}$$

Eine typische Space-time Produkt-Kurve, aufgetragen über der Arbeitsspeichergröße m für das LRU-Verfahren hat etwa den in Bild 2.3 angegebenen Verlauf.

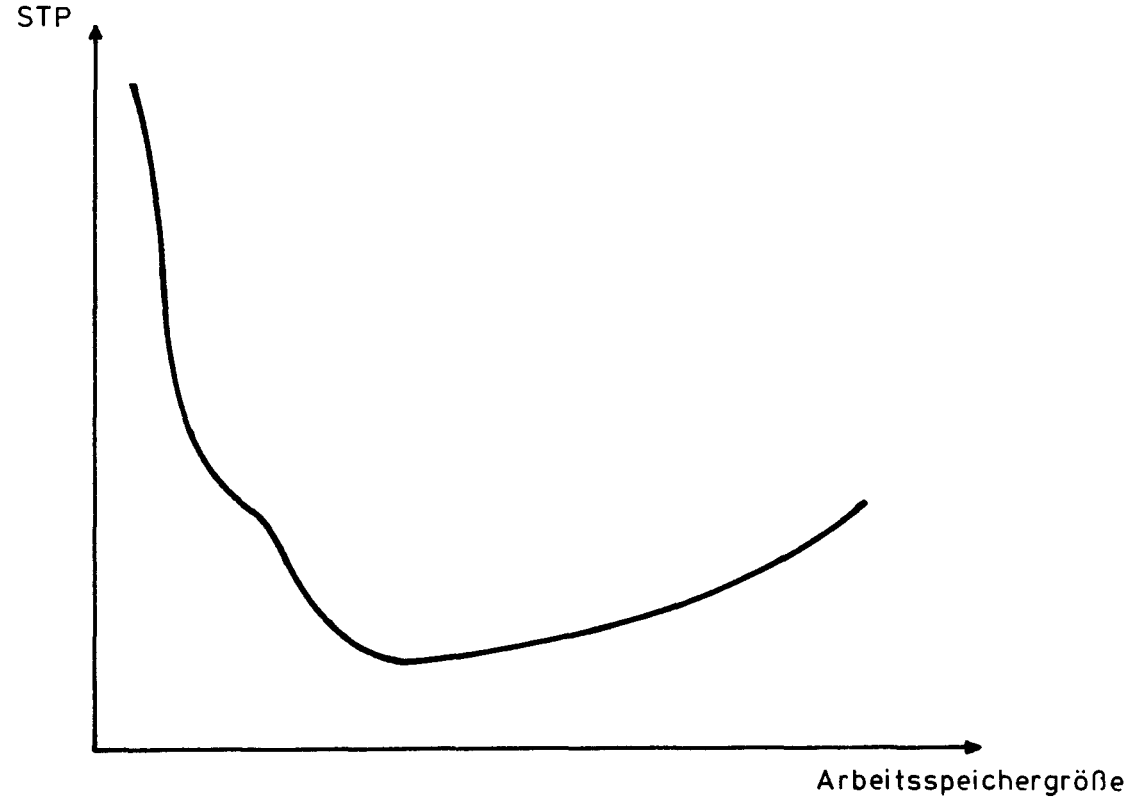


Bild 2.3
Space-time Produkt als Funktion der Arbeitsspeichergröße

Zum generellen Verlauf der Space-time Kurve ist zu sagen, daß man für kleine Werte von m wegen des überproportionalen Anstiegs der Fehlseitenrate ein steiles Ansteigen der Space-time Kurve beobachten kann; der Wert sinkt dann mit wachsender Arbeitsspeichergröße auf ein Minimum ab und steigt danach langsam an, weil von einer bestimmten Größe des Arbeitsspeichers an eine weitere Vergrößerung (space) nicht mehr durch eine adäquate Verringerung der Fehlseitenrate (time) kompensiert wird.

Für Verfahren mit fester Arbeitsspeichergröße ist das Space-time Produkt relativ einfach zu berechnen, weil nach der Aufbauphase die Arbeitsspeichergröße konstant gleich dem vorgegebenen Wert ist (und der Beitrag zum STP aus der Aufbauphase ist bezüglich des Seitenaustauschverfahrens eine additive Konstante).

Sinnvoller ist es jedoch, bei Verfahren fester Arbeitsspeichergröße nicht die Formel (2.1.5) zu benutzen, sondern dort und in (2.1.6) die zeitabhängige Größe $|S_t|$ durch m zu ersetzen, denn bei diesen Verfahren wird der gesamte Speicherbereich von m Seiten im allgemeinen vom Programmstart an zugeordnet, so daß er für andere Programme nicht verfügbar ist, selbst wenn das betrachtete Programm während der Aufbauphase nur einen Teil davon benötigt.

Bei Verfahren mit variabler Arbeitsspeichergröße ändert sich diese dynamisch und es müssen überdies die Zeitpunkte der Fehlseitenbedingungen exakt festgehalten werden, weil die Speichergrößen zu diesen Zeitpunkten besonders gewichtet werden (1 + Q statt 1).

Eine näherungsweise Berechnung des Space-time Produktes ist jedoch möglich. Sei

$$\overline{m}' = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} |S_t|$$

die mittlere Arbeitsspeicherbelegung in virtueller Zeit und

$$\overline{m}'' = \frac{1}{FS} \sum_{t=1}^{|\omega|} \Delta(t) \cdot |S_t|$$

die mittlere Arbeitsspeichergröße während der Ladezeiten, dann läßt sich die Formel für das Space-time Produkt wie folgt schreiben:

$$\begin{aligned} STP_R &= \overline{m}' \cdot |\omega| + FS \cdot \overline{m}'' \cdot Q \\ &= \overline{m}' \cdot |\omega| + FSR \cdot |\omega| \cdot \overline{m}'' \cdot Q \end{aligned} \qquad (2.1.7)$$

Unterstellt man nun, daß die mittlere Speichergröße während der Ausführungszeiten nicht wesentlich verschieden ist von der mittleren Speichergröße während der

Ladezeiten, so kann man näherungsweise $\overline{m}' = \overline{m}''$ setzen und erhält als Näherungswert für das Space-time Produkt

$$\widehat{STP}_R = \overline{m}' \, |\omega| \, (1 + FSR(\overline{m}') \cdot Q) , \qquad (2.1.8)$$

d.h. das Space-time Produkt kann näherungsweise aus der mittleren (virtuellen) Arbeitsspeichergröße und der Fehlseitenrate berechnet werden.

2.1.4 Prozessor - Intensität (duty factor)

Unter der Prozessor-Intensität versteht man die relative Prozessornutzung, d.h. den Quotienten aus der virtuellen Laufzeit eines Programmes und der realen Laufzeit, wobei hier - wie beim Space-time Produkt- nur diejenigen Verzögerungen, die durch vom Programm selbst verursachte Fehlseitenbedingungen hervorgerufen werden, bei der realen Zeit berücksichtigt werden. Wie die bereits vorher beschriebenen Performance-Maße ist auch die Prozessor-Intensität neben der Arbeitsspeichergröße (m) auch vom Seitenaustauschverfahren (A) und vom Programm (ω) abhängig.

$$DF(m) = \frac{|\omega|}{|\omega| + Q \cdot |\omega| \cdot FSR(m)} = \frac{1}{1 + Q \cdot FSR(m)} \qquad (2.1.9)$$

Das Performance-Maß Prozessor-Intensität hat bemerkenswerte Eigenschaften bei wechselnden Arbeitsspeichergrößen; es ist die mittlere Prozessor-Intensität gleich dem Mittel aus den Werten für die Prozessor-Intensität über Perioden fester Arbeitsspeicherzuordnung.

Seien m(1),..., m(k) die verschiedenen beim Programmablauf auftretenden Arbeitsspeichergrößen und L(1),..., L(k) und T(1),..., T(k) die jeweiligen Gültigkeitsdauern in virtueller bzw. realer Zeit, dann gilt für die durchschnittliche Prozessor-Intensität

$$\overline{DF} = \frac{L(1) + \ldots + L(k)}{T(1) + \ldots + T(k)} = \frac{1}{T} \sum_{i=1}^{k} L(i)$$

$$= \frac{1}{T} \sum_{i=1}^{k} \frac{L(i)}{T(i)} \cdot T(i)$$

$$= \frac{1}{T} \sum_{i=1}^{k} DF(m(i)) \cdot T(i) \qquad (2.1.10)$$

$$\text{mit} \quad T = \sum_{i=1}^{k} T(i).$$

Des weiteren ergibt die mittlere Prozessor-Intensität, indem in Gleichung (2.1.9) die Fehlseitenrate durch die mittlere Fehlseitenrate (virtuell) ersetzt wird:

$$\overline{DF} = \frac{\sum_{i=1}^{k} L(i)}{\sum_{i=1}^{k} T(i)} = \frac{\sum_{i} L(i)}{\sum_{i} L(i) \cdot (1 + Q \cdot FSR(m(i)))}$$

$$= \frac{\sum_{i} L(i)}{\sum_{i} L(i) + Q \cdot \sum_{i} L(i) \cdot FSR(m(i))}$$

$$= \frac{1}{1 + Q \dfrac{\sum_{i} L(i) \cdot FSR(i)}{\sum_{i} L(i)}}$$

$$= \frac{1}{1 + Q \cdot \overline{FSR}}. \qquad (2.1.11)$$

Die Prozessor-Intensität kann auch herangezogen werden, um die Prozessornutzung in Multiprogramming-Systemen abzuschätzen.

2.2. Demand Paging Verfahren fester Arbeitsspeichergröße

Bei Demand Paging Verfahren fester Arbeitsspeichergröße gilt für die Speicherzustände

$$S_t = \begin{cases} S_{t-1}, \text{ falls } r_t \in S_{t-1} \\ S_{t-1} + r_t, \text{ falls } r_t \notin S_{t-1} \text{ und } |S_{t-1}| < m \\ S_{t-1} + r_t - y_t, \text{ falls } r_t \notin S_{t-1} \text{ und } |S_{t-1}| = m. \end{cases}$$

Für die Menge der zu ladenden Seiten (X_t) und die Menge der zu ersetzenden Seiten (Y_t) gilt

$$|Y_t| \leq |X_t| \leq 1.$$

Die Ersetzungsstrategien können beschrieben werden durch die Angabe der zu ersetzenden Seite. Die zu ersetzende Seite wird im Folgenden mit $R_t(S_{t-1}, q_{t-1}, r_t)$ bezeichnet; dabei bezeichnet S_{t-1} den Arbeitsspeicherinhalt zum Zeitpunkt t - 1, q_{t-1} den zugehörigen Kontrollzustand und r_t die Seitenreferenz zum Zeitpunkt t. Unter dem Kontrollzustand sind die neben dem Speicherinhalt zur vollständigen Beschreibung eines Verfahrens gehörenden Zusatzinformationen (z.B. Alter einer Seite, Change-Bit, Referenz-Bit usw) zu verstehen.Die Ersetzungsfunktion R läßt sich für alle gängigen Verfahren relativ leicht angeben, es werden nachfolgend jedoch nur diejenigen Verfahren beschrieben, die bei den durchgeführten Simulationen verwendet werden.

2.2.1 Nichtoptimale Verfahren

Von den nichtoptimalen Verfahren hat sich das LRU (least recently used)-Verfahren im Laufe der Zeit als das deutlich Beste herauskristallisiert. Es wird deshalb als einziges Verfahren dieser Klasse zum Vergleich herangezogen und nachfolgend beschrieben.

LRU - Verfahren

Definition: In einer Referenzkette $\omega = r_1 r_2 \ldots r_t \ldots$ bezeichnet die Rückwärtsdistanz

$b_t(x)$ der zum virtuellen Zeitpunkt t referierten Seite x den virtuellen Zeitabstand zur letzten zurückliegenden Referenz auf die Seite x vor dem Zeitpunkt t:

$$b_t(x) = \begin{cases} k, \text{ falls } r_{t-k} = x \text{ und } r_{t-j} \neq x \text{ für alle } j = 1,\ldots,k-1 \\ \infty, \text{ falls } r_j \neq x \text{ für alle } j = 1, \ldots, t \ . \end{cases} \qquad (2.2.1)$$

Beim LRU - Verfahren wird diejenige Seite aus dem Arbeitsspeicher entfernt, die am längsten nicht mehr referiert wurde, d.h.

$$R_t(S_{t-1}, q_{t-1}, x) = y \text{ genau dann, wenn } b_t(y) = \max_{z \in S_{t-1}} [b_t(z)]$$

ist, wobei $r_t = x \notin S_{t-1}$ und $|S_{t-1}| = m$ vorausgesetzt ist.

Das LRU - Verfahren besitzt einige hervorragende Eigenschaften, was das Regularitätsverhalten angeht (vgl. Kap. 2.5).

2.2.2 Optimale Verfahren

Ein optimales Verfahren in der Klasse der Demand Verfahren fester Arbeitsspeichergröße ist das von BELADY /B4/ erstmals 1966 vorgestellte OPT-Verfahren, dessen Optimalität später /M2, C6/ nachgewiesen wurde; Optimalität bedeutet, daß die Fehlseitenrate dieses Verfahrens von keinem anderen Verfahren der Klasse unterschritten werden kann. Das Verfahren wird in der Literatur auch als B_0- oder MIN - Verfahren bezeichnet.

OPT - Verfahren

Definition: In einer Referenzkette $\omega = r_1 r_2 \ldots r_t \ldots$ bezeichnet die Vorwärtsdistanz $d_t(x)$ der zum virtuellen Zeitpunkt t referierten Seite x den virtuellen Zeitabstand bis zur nächstfolgenden Referenz der Seite x nach dem Zeitpunkt t:

$$d_t(x) = \begin{cases} k, \text{ falls } r_{t+k} = x \text{ und } r_{t+j} \neq x \text{ für alle } j = 1, \ldots, k-1 \\ \infty, \text{ falls } r_{t+j} \neq x \text{ für alle } j. \end{cases} \qquad (2.2.2)$$

Beim OPT - Verfahren wird diejenige Seite mit der größten Vorwärtsdistanz aus dem Arbeitsspeicher entfernt. Die Eindeutigkeit des Verfahrens wird durch lexikographische Anordnung der Seiten erreicht. Mehrdeutigkeit kann grundsätzlich nur auftreten, falls $d_t(x) = d_t(z)$ ist, was nur im Falle $d_t(x) = d_t(z) = \infty$ möglich ist, so daß die oben angegebene Vorschrift zur Erzielung der Eindeutigkeit keinen Einfluß auf die Performance des Verfahrens hat.

Beim OPT - Verfahren ist

$$R_t(S_{t-1}, q_{t-1}, x) = y \text{ genau dann, wenn } y = \min_{z \in S^*_{t-1}} [z] \text{ ist,}$$

wobei die Menge S^*_{t-1} genau aus der Teilmenge von S_{t-1} besteht, für deren Elemente

$$d_t(z) = \max_{u \in S_{t-1}} [d_t(u)]$$

gilt.

2.3 Demand Paging Verfahren variabler Arbeitsspeichergröße

GRAHAM /G3/ unterteilt die Verfahren variabler Arbeitsspeichergröße in drei Klassen:

(1) Die Größendynamik ist nicht durch die Programmeigenschaften gesteuert.

(2) Die Größendynamik ist teilweise oder ganz durch die Programmeigenschaften gesteuert, aber der Working Set ist nicht geschützt.

(3) Die Größendynamik ist programmgesteuert und der Working Set eines jeden rechenbereiten oder wegen Page - I/O suspendierten Programms ist geschützt, so daß sich eine automatische Steuerung des Multiprogramming-Grades ergibt.

Die Zuordnung realer Verfahren zu den Klassen ist nicht immer eindeutig. In die erste Klasse fallen Verfahren, die sehr eng an Verfahren fester Arbeitsspeichergröße angelehnt sind, bei denen die feste Speichergröße nach gewissen Zeitintervallen (etwa im Zuge einer vorübergehenden bevorzugten Ausführung) verändert wird.

Eine weitere Klasse von Verfahren, die zwischen (1) und (2) einzuordnen ist, entsteht durch globale Anwendung der bekannten Verfahren fester Arbeitsspeichergröße auf den gesamten realen Speicher. Nimmt man als Beispiel LRU - Strategie an, so ist klar, daß die Seiten des jeweils zuletzt aktiven Programms - weil es die zuletzt referierten sind - bevorzugt vor den Seiten der anderen Programme im Arbeitsspeicher gehalten werden, woraus sich auf das einzelne Programm bezogen eine Dynamik in der Größe des zugeordneten Speichers ergibt. Es liegt auf der Hand, daß die Qualität dieser Verfahren sehr stark vom Zuteilungsverfahren der Zentraleinheit (sheduling) abhängig ist, da bei einem unpassenden Zuteilungsverfahren der Bezug von Arbeitsspeicherinhalt und -größe zum Programmverhalten verloren gehen kann, weil die Seiten eines Programmes möglicherweise nicht deshalb aus dem Arbeitsspeicher entfernt werden, weil sie nicht mehr zur Aktivitätsmenge gehören, sondern weil das Programm aufgrund einer relativen Benachteiligung beim Zuteilungsverfahren keine Gelegenheit hatte, die Seiten seines Working Set zu referieren.

Alle Verfahren der Klasse (2) sind in dem Sinne globale Verfahren, als die Speicherplatzzuteilung abhängig ist von der Anzahl der gleichzeitig in Ausführung befindlichen Programme und deren Speicherplatzanforderungen, die ihrerseits stark abhängig sind vom Zuteilungsverfahren für die CPU (worunter auch die Steuerung des Multiprogramming-Grades fällt).

Die Verfahren der Klasse (3) können insofern als 'lokale' Verfahren bezeichnet werden, als Anzahl und Auswahl der im Arbeitsspeicher gehaltenen Seiten eines Programmes nur vom Programm selbst abhängen. Dies wird durch die Bedingung erreicht, daß sich nur solche Programme um die Zentraleinheit bewerben können, deren Working Set sich komplett im Arbeitsspeicher befindet. Bei der Konstruktion der Unabhängigkeit der Programme voneinander ist es ausgeschlossen, daß ein Programm Seitenrahmen eines anderen in den Arbeitsspeicher geladenen Programmes bekommen kann, was bei fester Gesamtgröße des Arbeitsspeichers zu einer verfahrensabhängigen Steuerung des Multiprogramming-Grades führt.

Konkret realisiert wird die rückwirkungsfreie (was den Working Set angeht) dynamische Speicherplatzzuordnung durch die Existenz eines Pools nicht zugeteilter Seitenrahmen, der - falls er erschöpft sein sollte - durch Ausladen (Swappen) eines Programmes wieder aufgefüllt wird. Bezüglich der Performance sind die Verfahren der Klasse (3) den Verfahren der Klassen (1) und (2) überlegen; außerdem sind sie weniger empfindlich gegenüber instabilen Systemzuständen als etwa Verfahren der Klasse (2).

Verfahren der Klasse (3) sind das DWS (Denningsches Working Set)-Verfahren und das PFF (page fault frequency)-Verfahren sowie auch das optimale VMIN - Verfahren, die nachfolgend beschrieben werden.

2.3.1 Nichtoptimale Verfahren

DWS - Verfahren

Bei dem von DENNING /D1, D2/ vorgestellten Working Set Verfahren wird eine Seite

nach genau T virtuellen Zeiteinheiten dauerndem unreferierten Aufenthalt aus dem Arbeitsspeicher entfernt.

Der Working Set W(T,t) zum Zeitpunkt t enthält die Menge der verschiedenen Seiten, die im Abschnit [t - T + 1, t] der Referenzkette ω referiert wurden. Der Verfahrensparameter T wird auch Fenstergröße genannt, weil man sich zur Bestimmung des Working Set ein mit dem Programmablauf (d.h. t) über die Referenzkette ω gleitendes Fenster der Größe T vorstellen kann, wobei die innerhalb des durch das Fenster gegebenen Abschnittes der Referenzkette referierten Seiten den jeweiligen Working Set bilden.

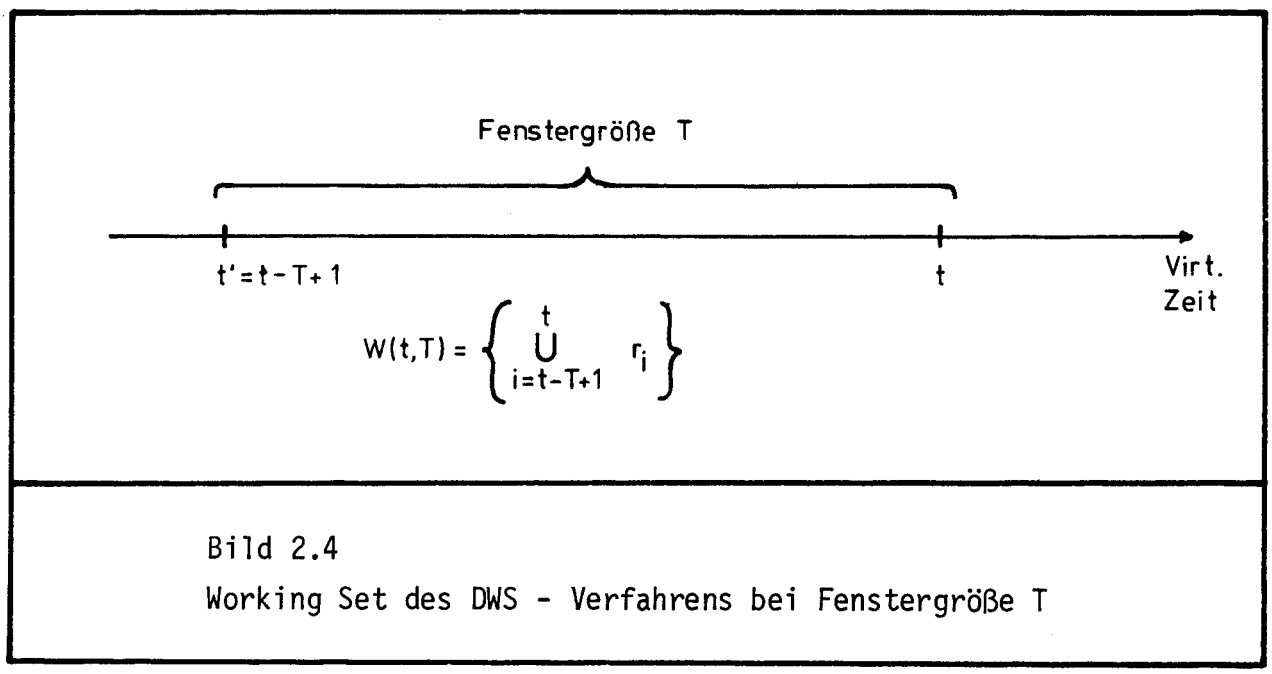


Bild 2.4
Working Set des DWS - Verfahrens bei Fenstergröße T

Einige Eigenschaften des Working Set sind aus der Definition sofort ableitbar.

(1) $W(0,T) = \emptyset$ für alle T,

(2) $W(t, T)$ ist definiert für $T \geq 1$.

(3) Für $t < T$ gilt $W(t, T) = \{ \bigcup_{l=1}^{t} r_l \}$ (Aufbauphase des WS).

(4) Es gilt $W(t, T) \subseteq W(t, T + 1)$.

Die mittlere Working Set-Größe w (T) ist gegeben durch

$$w(T) = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} w(t, T) \text{ mit } \quad w(t, T) = |W(t, T)| \ .$$

Aufgrund einer Reihe von sehr positiven Eigenschaften (vgl. etwa /D6, G3/) ist das DWS - Verfahren Gegenstand intensiver Studien gewesen, obwohl das Verfahren so aufwendig ist, daß eine Implementation nur bei entsprechender Hardware - Unterstützung - wie sie etwa beim MANIAC II-Projekt /M3/ gegeben war - sinnvoll ist.

Es besteht eine sehr enge Verwandtschaft zwischen dem DWS- und dem LRU-Verfahren; man kann das DWS - Verfahren als eine bzgl. des Speicherplatzbedarfs dynamisierte Variante des LRU - Verfahrens betrachten. Beide Verfahren entfernen die am längsten nicht mehr referierte Seite aus dem Arbeitsspeicher; beim DWS - Verfahren geschieht dies nach einer vorgegebenen unreferierten Verweildauer, beim LRU - Verfahren nach nicht vorhersehbarer Zeit bei Bedarf eines freien Seitenrahmens, so daß beim DWS - Verfahren die maximale unreferierte Verweildauer einer Seite im Arbeitsspeicher fest und die Größe des belegten Arbeitsspeichers variabel ist, während beim LRU - Verfahren die Arbeitsspeicherbelegung fest und die maximale unreferierte Verweildauer variabel ist. Würde man beim DWS-Verfahren die Fenstergröße T so variieren, daß für alle t w(t, T) = const. =m ist, dann wären die Speicherzustände identisch mit denen des LRU-Verfahrens.

Ebenso besteht auch eine sehr enge Verwandtschaft zwischen dem DWS-Verfahren und dem in Kap. 2.3.2 beschriebenen VMIN-Verfahren.

Betrachtet man die Dynamik der Working Set-Größe, so ist diese zum einen durch relativ lang dauernde Einbrüche, die zweifellos die Folge kleiner hoch iterativer Programmabschnitte sind, gekennzeichnet, zum anderen aber auch wiederkehrende relativ kurz dauernde Expansionen. Letzteres Phänomen wird einsichtig, wenn man - wie es durch MADISON et al. /M1/ und GRAHAM /G3/ nahegelegt wird - für das

Programmverhalten ein Phasen-/Übergangsmodell unterstellt: An der Übergangsstelle zweier Programmphasen gehören kurzfristig die Lokalitäten beider Phasen oder Teile davon dem Working Set an und werden deshalb im Arbeitsspeicher gehalten. Der Verlauf der Working Set-Größe ist in Bild 2.5 dargestellt; d(i) ist die Dauer eines Referenzzyklus aller zur Phase i gehörenden Seiten.

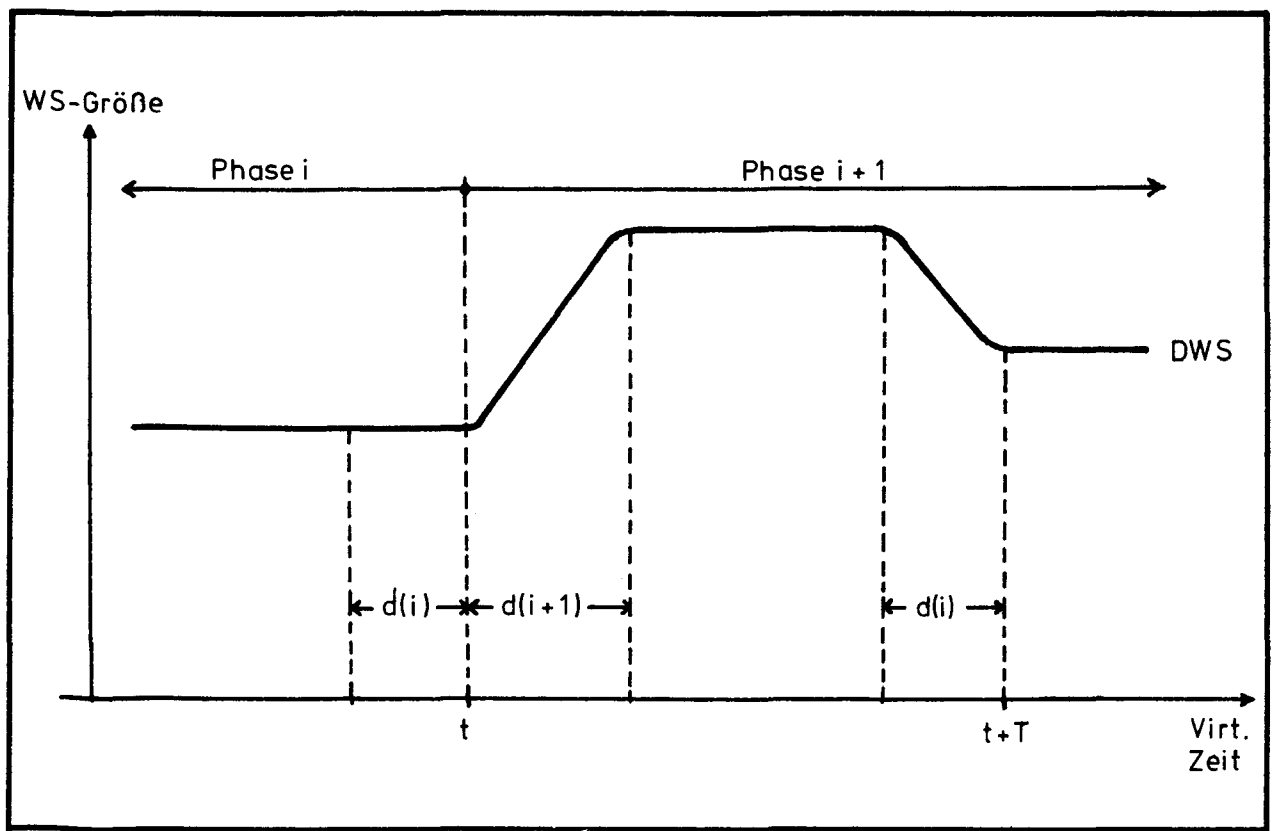


Bild 2.5
Verhalten des DWS - Verfahrens an Phasenübergängen

Verbunden ist dieses Verhalten an Phasenübergangsstellen mit einer erhöhten Fehlseitenrate, die die Ursache dafür ist, daß der Working Set Parameter T nur schlecht geeignet ist, um die Fehlseitenrate zu steuern: Innerhalb einer Phase deutet eine erhöhte Fehlseitenrate darauf hin, daß die Fenstergröße kleiner ist als die Dauer eines Referenzzyklus ($T < d(i)$) und eine Vergrößerung von T wäre die angemessene

Reaktion; bei der erhöhten Fehlseitenrate an Phasenübergangsstellen jedoch müßte T verkleinert werden, um die kurzfristige Expansion des Working Set gering zu halten.

Die Performance des DWS-Verfahrens hängt stark von der Wahl des Parameters T ab. Generell muß die Wahl von T unter drei Gesichtspunkten gesehen werden:

(1) T muß ausreichend groß sein, so daß die Dauer eines Referenzzyklus der aktuellen Programmphase mit großer Wahrscheinlichkeit kleiner als T ist (oder äquivalent, daß alle Seiten der aktuellen Lokalität im Working Set enthalten sind).

(2) T sollte nicht wesentlich größer sein als im Mittel die Gültigkeitsdauer einer Programmphase, so daß die Wahrscheinlichkeit, daß mehr als ein Phasenübergang im Working Set enthalten ist, gering ist.

(3) Da über T implizit die Fehlseitenrate gesteuert wird, muß T auch im Zusammenhang mit der Zugriffszeit zum Sekundärspeicher gesehen werden, d. h. T ist so groß zu wählen, daß durch die daraus resultierende Fehlseitenrate die Leistungsfähigkeit des Sekundärspeichers nicht überfordert wird.

Besonders die Punkte (1) und (2) sind nicht leicht zu verwirklichen, da die herangezogenen Kriterien auf Programmeigenschaften Bezug nehmen, die dynamisch sehr stark schwanken können. So lautet ein Vorschlag von DENNING selbst /D1/, explizit bezugnehmend auf Kriterium (3), daß für T die doppelte Ladezeit einer Seite gewählt werden solle.

Zahlreiche Untersuchungen haben gezeigt, daß bei geeigneter Wahl von T der Working Set des DWS-Verfahrens eine gute Schätzung der Programmlokalität liefert und daß es durchaus möglich ist, allgemeingültige T-Werte zu finden, so daß das Verfahren in fast allen Fällen mit nahe-optimaler oder zumindest doch akzeptabler Performance

arbeitet.

PFF - Verfahren

Ein zweites, dem DWS-Verfahren vergleichbares Verfahren ist das von CHU und OPTERBECK /C3, O1/ vorgeschlagene und untersuchte PFF-Verfahren. Bei diesem Verfahren werden Schätzungen der Programmlokalität nur beim Auftreten einer Fehlseitenbedingung vorgenommen. Das PFF-Verfahren benutzt den in virtueller Zeit gemessenen Abstand von der letzten zurückliegenden Fehlseitenbedingung als Fenstergröße und ist somit wie das DWS-Verfahren ein Verfahren, das mit einem (über die Referenzkette) bewegten Fenster arbeitet, jedoch im Gegensatz zu diesem mit einem Fenster variabler Größe. Hierbei wird die Arbeitsspeicherzuordnung durch den zeitlichen Abstand aufeinanderfolgender Fehlseitenbedingungen gesteuert. (Der Kehrwert kann als Frequenz interpretiert werden, daher der Name des Verfahrens.)

Seien t', $t(t' < t)$ Zeitpunkte aufeinander folgender Fehlseitenbedingungen, dann ist der Arbeitsspeicherinhalt zum Zeitpunkt t bestimmt durch

$$S(t, T) = \begin{cases} W(t, t - t') & \text{für} \quad t - t' > T \\ S(t', T) + \{r_t\} & \text{für} \quad t - t' \leq T \end{cases} \qquad (2.3.1)$$

mit einem vorgegebenen Zeitparameter T.

$t - t' > T$ wird so interpretiert, daß genügend Zeit war, um alle zur aktuellen Programmlokalität gehörenden Seiten zu referieren, und deshalb werden alle Seiten, die im Intervall $(t', t]$ nicht referiert wurden als nicht zur gültigen Lokalität gehörend betrachtet und aus dem Arbeitsspeicher entfernt.

$t - t' \leq T$ wird so interpretiert, daß das Programm nicht genügend Zeit hatte, alle zur aktuellen Lokalität gehörenden Seiten zu referieren, so daß eine Überprüfung des Speicherinhaltes nicht sinnvoll ist, weil aus der Nichtreferenz einer Seite nicht geschlossen werden kann, daß sie nicht zur gültigen Lokalität gehört. Gleichzeitig

zeigt der kurze Zeitabstand aufeinander folgender Fehlseitenbedingungen (= hohe Fehlseitenrate) an, daß dem Programm nicht genügend Arbeitsspeicher zur Verfügung steht; deshalb wird dem Programm für die Page Fault verursachende Seite ein zusätzlicher Seitenrahmen zur Verfügung gestellt.

Im Gegensatz zum DWS-Verfahren ist das PFF-Verfahren mit geringem Aufwand zu realisieren. Es bedarf eines Referenzbits pro Seitenrahmen, das nach jedem Auftreten einer Fehlseitenbedingung für alle Seiten des Working Set gelöscht und beim Zugriff auf eine Seite (i.a. hardwaremäßig) gesetzt wird, und eines Zeitgebers, der die Zeitabstände aufeinander folgender Fehlseitenbedingungen mißt, so daß diese mit T verglichen werden können.

Von der Performance her ist das PFF-Verfahren dem DWS-Verfahren leicht unterlegen, da es bei gleicher Fehlseitenrate durchweg eine etwas höhere mittlere Arbeitsspeicherbelegung produziert. Dies wird einsichtig, wenn die Art und Weise der Working Set-Bildung im Zusammenhang mit dem bereits mehrfach verwendeten Phasen-/Übergangsmodell des Programmverhaltens gesehen wird. Während nämlich das DWS-Verfahren - wie bereits beschrieben - nur dazu tendiert, kurzfristig an den Phasenübergangsstellen zwei Lokalitäten (i.a. sogar nur Teile davon) im Arbeitsspeicher zu halten, tendiert das PFF-Verfahren dazu, grundsätzlich zwei Lokalitäten in seinem Working Set zu halten (außer an den Phasenübergangsstellen, wo es sich ähnlich wie das DWS-Verfahren verhält).

Das Verhalten des PFF-Verfahrens bezüglich der Speicherbelegung ist in Bild 2.6 dargestellt. Sei t_i der Zeitpunkt des Phasenüberganges von der Phase i - 1 zur Phase i und Zeitpunkt der ersten Fehlseitenbedingung zum Aufbau der Lokalität L(i). Bei vernünftiger Wahl des Parameters T finden während der Aufbauphase keine Arbeitsspeicherüberprüfungen statt. (Es würden sonst mit hoher Wahrscheinlichkeit auch Seiten der gerade im Aufbau befindlichen Lokalität aus dem Arbeitsspeicher entfernt). Während der Gültigkeitsdauer der Phase i (dies gilt für jede Phase gemäß

Def.) tritt bis zum Übergang zur Phase $i + 1$ an der Stelle t_{i+1} keine weitere Fehlseitenbedingung auf. Da T kleiner sein sollte als die Gültigkeitsdauer einer Phase, wird zu diesem Zeitpunkt eine Überprüfung des Working Set vorgenommen, wobei alle Seiten, die nach der Komplettierung von L(i) nicht mehr referiert wurden, aus dem Arbeitsspeicher entfernt werden; dies sind nach dem Verständnis einer Programmphase nicht die Seiten von L(i), sondern die bis zu diesem Zeitpunkt noch im Arbeitsspeicher befindlichen Seiten von L(i - 1), so daß dann nur noch die Seiten von L(i) (+ der ersten Fault verursachenden Seite von L(i + 1)) zum Working Set gehören. Beim anschließenden Aufbau der Lokalität L(i + 1) kommen die Seiten dieser Lokalität hinzu, so daß sich nach der Aufbauphase L(i) U L(i + 1) im Arbeitsspeicher befindet.

Zum Vergleich ist in Bild 2.6 auch das Größenverhalten des Working Sets des DWS-Verfahrens eingezeichnet.

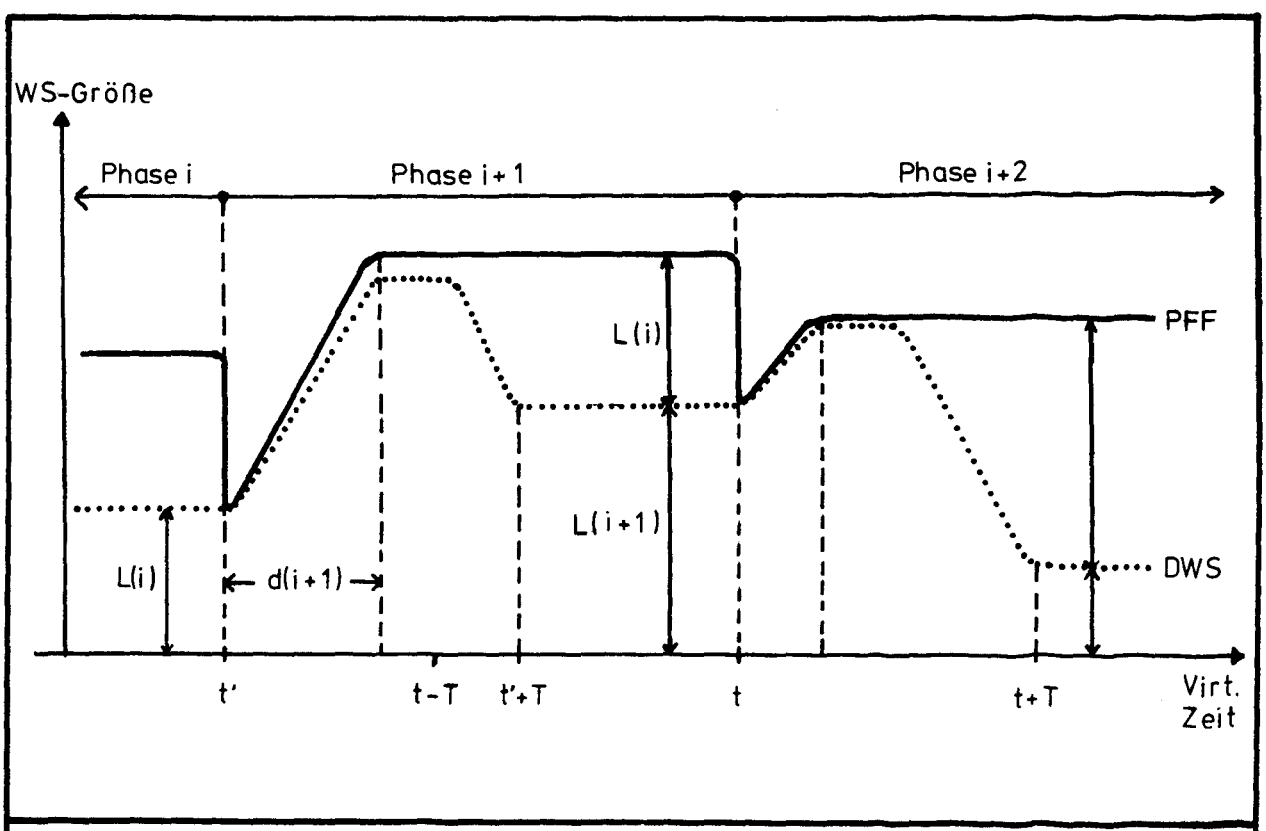


Bild 2.6
Verhalten des PFF - Verfahrens an Phasenübergängen

Das PFF-Verfahren ist später (dies ist nicht Bestandteil der ursprünglichen Definition des Verfahrens von CHU und OPDERBECK) erweitert worden durch einen zweiten Verfahrensparameter T'. T' kann interpretiert werden als maximaler Zeitabstand für eine Überprüfung des Working Set. Ein gewisser Nachteil des nicht modifizierten Verfahrens ist, daß bei großem Abstand aufeinander folgender Fehlseitenbedingungen (d.h. bei langer Gültigkeitsdauer einer Phase) nicht mehr zur Aktivitätsmenge gehörende Seiten unnötig lange zum Working Set gehören. Im Extremfall könnten während einer Aufbauphase in kurzem Abstand ($< T$ Zeiteinheiten) alle Programmseiten referiert worden sein; da in diesem Falle weitere Fehlseitenbedingungen nicht mehr auftreten können, würden ohne die Zeitbegrenzung durch T' alle Programmseiten während der gesamten Programmlaufzeit im Arbeitsspeicher gehalten werden.

Zur Realisierung dieser Variante des Verfahrens wird ein zweiter Zeitgeber benötigt, aufgrund dessen Wirksamwerden nach T' Zeiteinheiten die gleiche Überprüfungsaktion (inkl. Löschen des Referenzbits) durchgeführt wird wie beim Auftreten einer Fehlseitenbedingung.

Wie beim DWS-Verfahren müssen auch die Parameter des PFF-Verfahrens unter verschiedenen Gesichtspunkten gesehen werden. Relativ unkritisch ist der Parameter T' , der im wesentlichen der Verhinderung eines extrem ungünstigen Verfahrensverhaltens dient und im allgemeinen, da er zusätzliche Programmunterbrechungen produziert, groß gewählt wird.

An den Parameter T sind drei Forderungen zu stellen, die z.T. schon angeklungen sind:

(1) T sollte ausreichend groß sein, so daß ein ungestörter Aufbau einer Programmlokalität möglich ist; dies ist nur möglich, wenn $t - t' \leq T$ ist, weil bei einer Speicherüberprüfung mit hoher Wahrscheinlichkeit auch Seiten der im Aufbau befindlichen Lokalität aus dem Arbeitsspeicher entfernt würden.

(2) T sollte im Mittel kleiner als die Gültigkeitsdauern der Programmphasen

sein, damit mit hoher Wahrscheinlichkeit nicht mehr als die Seiten zweier benachbarter Programmphasen zum Working Set gehören.

(3) Wie auch beim DWS-Verfahren wird über den Verfahrensparameter die mittlere Arbeitsspeicherbelegung und die Fehlseitenrate beeinflußt, so daß die Parameterwahl auch im Kontext mit der Leistungsfähigkeit des Sekundärspeichers gesehen werden muß.

Anders als beim DWS-Verfahren sind konkrete Angaben bzgl. der Wahl des PFF-Verfahrensparameters dem Verfasser nicht bekannt, er ist jedoch kleiner als der DWS-Parameter zu wählen.

2.3.2 Optimale Verfahren

Performance Messungen haben gezeigt, daß das OPT-Verfahren häufig schlechter abschneidet als die nichtoptimalen Verfahren mit variabler Arbeitsspeichergröße, also kein optimales Verfahren dieser Klasse sein kann.

Bei der Suche nach einem optimalen Verfahren variabler Speichergröße ist zunächst ein geeignetes Optimalitätskriterium zu ermitteln. Es ist klar, daß die Minimierung der Fehlseitenrate wie beim OPT-Verfahren kein geeignetes Kriterium bei Verfahren variabler Speichergröße ist, denn diese wird zweifellos minimal, wenn das gesamte Programm in den Arbeitsspeicher geladen wird. Ein geeignetes Performance-Maß für die Bewertung von Verfahren variabler Arbeitsspeichergröße ist, weil sowohl die Speicherbelegung als auch die Anzahl der Page Faults berücksichtigt werden, das Space - time Produkt. Leider existiert kein einfaches Verfahren, das zu einer Minimierung des in Realzeit gemessenen Space - time Produktes führen würde. Ein entsprechendes Verfahren ist formuliert worden /B7/, es ist jedoch so aufwendig, daß es auch für Simulationen praktisch nicht in Frage kommt. Hinzu kommt, daß ein solches Verfahren eine für die Praxis sehr unerfreuliche Eigenschaft hätte: es zeigt einen gewissen Selbstverstärkungseffekt bzgl. der Fehlseitenhäufigkeit. Häufige Fehlseitenbedingungen bedeuten, daß die Interreferenzintervalle (d.h. die Abstände von Referen-

zen der gleichen Seite) in Realzeit lang werden und damit die Wahrscheinlichkeit steigt, daß Seiten aus dem Arbeitsspeicher entfernt werden, die nachfolgend wieder geladen werden müssen; somit zeigt das Verfahren die Tendenz, das Space - time Produkt nicht durch eine Reduktion der Anzahl der Page Faults, sondern durch Verkleinerung des Schadens je Page Fault zu minimieren.

Infolgedessen hat sich als optimales Verfahren das von PRIEVE und FABRY /P4/ vorgeschlagene VMIN-Verfahren durchgesetzt, bei dem eine Kostenfunktion minimiert wird, die große Ähnlichkeit mit der Formel für das Space - time Produkt aufweist, die jedoch auf virtuellen Zeiten basiert, was eine seitenweise, lokale Kostenminimierung gestattet.

VMIN – Verfahren

Der Grundgedanke beim VMIN-Verfahren ist, die Verarbeitung einer Referenzkette kostenoptimal abzuwicklen, indem nach jeder Referenz die Kosten eines Ladevorganges abgewogen werden gegen die Kosten, die ein Verbleiben der soeben referierten Seite im Arbeitsspeicher bis zur nächstfolgenden Referenz auf diese Seite verursachen würde, wobei das Interreferenzintervall in virtueller Zeit gemessen wird.

Sei x die zum Zeitpunkt t referierte Seite und bezeichne U die Kosten, die entstehen, um eine Seite für eine (virtuelle) Zeiteinheit im Arbeitsspeicher zu halten, dann sind die Kosten, die entstehen, wenn die Seite x bis zur nächsten Referenz im Arbeitsspeicher gehalten wird, gegeben durch

$$C_t^{(1)} = d_t(x) \cdot U \qquad (d_t(x) = \text{Vorwärtsdistanz, vgl. (2.2.2)})$$

und die Kosten, die entstehen, wenn die Seite x unmittelbar nach der Referenz zum Zeitpunkt t aus dem Arbeitsspeicher entfernt wird und zum Zeitpunkt $t + d_t(x)$ erneut geladen wird, gegeben durch

$$C_t^{(2)} = R \ ,$$

wobei R die Ladekosten für eine Seite bezeichnet.

Setzt man

$$\frac{R}{U} = T ,$$

so ist

$$C^{(1)} = d_t(x) \cdot U$$

und

$$C_t^{(2)} = T \cdot U.$$

Das VMIN-Verfahren entfernt die Seite x unmittelbar nach der Referenz zum Zeitpunkt t aus dem Arbeitsspeicher, wenn $d_t(x) > T$ ist, d.h. die Kosten des Speicheraufenthaltes die Ladekosten übersteigen. Umgekehrt formuliert, enthält der Working Set des VMIN-Verfahrens alle Seiten eines in die Zukunft gerichteten Fensters der Länge T, das über die Referenzkette gleitet. Das VMIN-Verfahren arbeitet also wie das DWS-Verfahren mit einem bewegten Fenster fester Größe, wobei das VMIN-Fenster allerdings im Gegensatz zum Fenster des realisierbaren DWS-Verfahrens in die Zukunft weist.

Die Optimalität des VMIN-Verfahrens bezüglich der schon erwähnten Kostenfunktion läßt sich relativ leicht beweisen. Die Gesamtkosten der Abarbeitung einer Referenzkette setzen sich zusammen aus den Kosten für die Speicherbelegung - hier gegeben durch das Space - time Produkt in virtueller Zeit mal Kosten pro Zeiteinheit und Seite (U) - und den Ladekosten - gegeben durch Anzahl Fehlseitenbedingungen mal Ladekosten pro Seite (R) - , d.h.

$$C = FSR \cdot |\omega| \cdot R + |\omega| \cdot \overline{m} \cdot U.$$

Diese Kosten können aber - was die Minimierung der Kostenfunktion erleichtert - auch folgendermaßen beschrieben werden:

$$C = n \cdot R + \sum_{t=1}^{|\omega|} c_t \quad (n = \text{Anzahl der Programmseiten}),$$

wobei $n \cdot R$ die (unvermeidlichen) initialen Ladekosten für die Programmseiten angibt und die c_t für alle Seiten die Kosten für die nachfolgenden Interreferenzintervalle beschreiben.

$$c_t = \begin{cases} U \cdot d_t, & \text{falls die Seite } r_t \text{ bis zur nächsten Referenz im Arbeitsspeicher bleibt,} \\ U \cdot z_t, & \text{falls } r_t \text{ überhaupt nicht mehr referiert wird und } z_t \text{ Zeiteinheiten bis zur Freigabe des Seitenrahmens vergehen,} \\ U \cdot z_t + R, & \text{falls die Seite nach } z_t \text{ Zeiteinheiten aus dem Speicher entfernt und anschließend wieder referiert wird.} \end{cases}$$

Wie leicht zu sehen ist, werden die c_t's durch die Verfahrensbedingungen des VMIN-Verfahrens minimiert; da die Entscheidungen unabhängig voneinander sind (was bei Messung in Realzeit nicht der Fall wäre!) wird durch Minimierung der einzelnen c_t's auch die Summe minimal.

Ins Auge fallend ist die enge Verwandtschaft zwischen dem VMIN- und dem DWS-Verfahren. Nicht nur, daß beide Verfahren gleichermaßen Zeitfensterverfahren sind, sie legen darüber hinaus bei übereinstimmendem Verfahrensparameter T ein identisches Verhalten bezüglich der Erzeugung von Fehlseitenbedingungen an den Tag (d.h. die gleichen Seiten erzeugen an identischen Zeitpunkten Fehlseitenbedingungen); dabei liefert das VMIN-Verfahren eine geringere mittlere Arbeitsspeicherbelegung, weil hier eine innerhalb der nächsten T Zeiteinheiten nicht mehr referierte Seite sofort aus dem Arbeitsspeicher entfernt wird und nicht erst, nachdem sie T Zeiteinheiten unnötigerweise (aber aufgrund des Kenntnisstandes beim DWS-Verfahren unvermeidlich) im Arbeitsspeicher geblieben ist. Aufgrund der Vorabkenntnis des Programmverlaufes kommt es beim VMIN-Verfahren nicht zu der für das DWS-Verfahren charakteristischen kurzfristigen Ausweitung des Working Set, sondern im Gegenteil zu einem kurzfristigen Einbruch; der Verlauf ist in Bild 2.7 dargestellt.

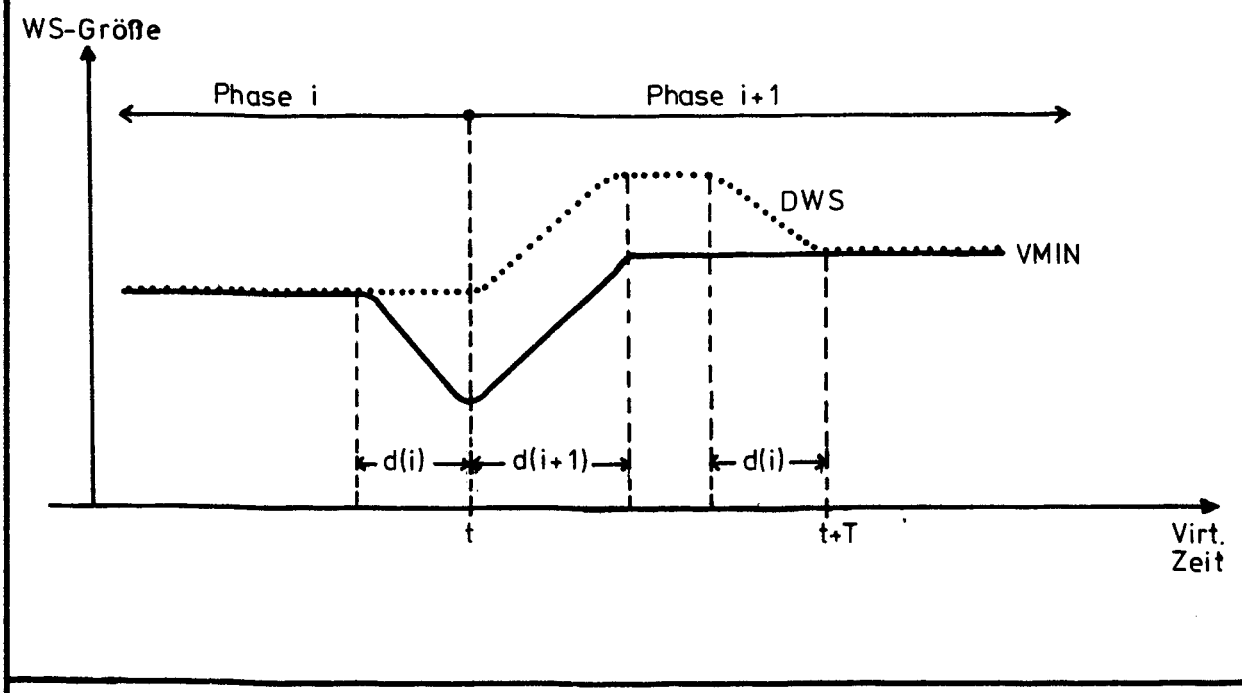


Bild 2.7
Verhalten des VMIN - Verfahrens an Phasenübergängen

Trivialerweise gilt, da das VMIN-Verfahren ein Zeitfensterverfahren mit fester Fenstergröße ist, für den Working Set bzgl. des Parameters T die Einschlußeigenschaft:

$$W(t, T) \subseteq W(t, T + 1) .$$

2.4 Realisierungsaufwand und abgeleitete Verfahren

Von den bekannten Algorithmen sind die optimalen nicht realisierbar, weil sie Informationen benötigen, die in der Praxis nicht zur Verfügung stehen. Von den interessantesten realisierbaren Verfahren - das sind LRU-, DWS- und PFF-Verfahren - ist nur das PFF-Verfahren so einfach, daß es exakt gemäß Definition in der Praxis verwirklicht werden kann. LRU- und DWS-Verfahren können, wenn man vom MANIAC-II-Projekt /M3/, bei dem das DWS-Verfahren mit Unterstützung spezieller Hardware in fast reiner Form realisiert wurde, absieht, aus Aufwandsgründen nur vergröbert realisiert werden.

Ein interessanter Mechanismus, mit dem sich sowohl LRU-artige als auch DWS-artige Algorithmen realisieren lassen, ist der im MVS zur Anwendung kommende Alterungsmechanismus /C2/. Dabei sind jedem Seitenrahmen des realen Arbeitsspeichers ein Referenzbit (RB) und ein UIC (=unreferenced interval counter) - genannter Zähler zugeordnet. RB wird hardwaremäßig gesetzt, sobald eine Adresse des zugeordneten Seitenrahmens referiert wird. Das Referenzbit wird periodisch nach I Zeiteinheiten abgeprüft und in Abhängigkeit des Zustandes werden folgende Aktionen getätigt:

$$RB = 1 \Longrightarrow RB := 0 \text{ und } UIC := 0$$

$$RB = 0 \Longrightarrow UIC := UIC + 1 .$$

Hieraus läßt sich ein LRU-artiges Verfahren ableiten, wenn in einem Arbeitsspeicher fester Größe bei Bedarf eine Seite mit maximalem UIC-Wert ersetzt wird. Hierbei wird die durch den LRU-Stack gegebene Vollordnung aller Programmseiten (vgl. Kap. 2.5) durch eine Klassenbildung ersetzt, wobei die Klassenelemente gleichwertig behandelt werden. Eine DWS-Strategie ergibt sich, wenn alle Seiten, deren UIC einen vorgegebenen Grenzwert G überschreitet, als nicht mehr zum Working Set gehörend betrachtet werden. Die Differenz zum DWS-Verfahren besteht darin, daß Seiten nicht genau $T = G \cdot I$ Zeiteinheiten nach ihrer letzten Referenz aus dem Arbeitsspeicher

entfernt werden, sondern nach zwischen $G \cdot I$ und $(G + 1) \cdot I$ Zeiteinheiten. A. RAFII /R1/ beschreibt ein modifiziertes DWS-Verfahren, das er MWS (= modified working set)-Verfahren nennt, bei dem der Verfahrensparameter T die Bedeutung hat, daß nach T (virtuellen) Zeiteinheiten der Working Set überprüft wird und alle Seiten, die während der letzten T Zeiteinheiten nicht referiert worden sind, aus dem Arbeitsspeicher entfernt werden. Dieses Verfahren ist ein Sonderfall des vorher beschriebenen Verfahrens (für $T = I$ und $G = 1$). Beim MWS-Verfahren beträgt die maximale referenzfreie Aufenthaltsdauer einer Seite im Arbeitsspeicher zwischen T und 2 T Zeiteinheiten gegenüber genau T Zeiteinheiten beim DWS-Verfahren, was zu einer höheren mittleren Arbeitsspeicherbelegung bei reduzierter Fehlseitenrate verglichen mit dem DWS-Verfahren bei gleichem Parameterwert T führt. Es wird gezeigt, daß die Performance des MWS-Verfahrens nicht wesentlich schlechter als diejenige des DWS-Verfahrens ist.

Ein anderer Ansatz zu einer noch weitergehenden Vereinfachung ergibt sich, wenn man für die Überprüfung der Arbeitsspeicherinhalte bei DWS-artigen bzw. der Klassifizierung der Seiten bei LRU-artigen Verfahren keinen unabhängigen Zeittakt, sondern die Zeitpunkte des Auftretens von Fehlseitenbedingungen als natürliche Überprüfungszeitpunkte verwendet. Bei Verfahren vom Typ 'LRU' wird dies so gehandhabt, daß die im Arbeitsspeicher befindlichen Seiten anhand der Referenzbits in zwei Klassen eingeteilt werden: Seiten, die seit dem Zeitpunkt der letzten Fehlseitenbedingung referiert wurden, und Seiten, die nicht referiert wurden. Natürlich ist auch hier eine Unterteilung in mehrere Klassen durch Zählung der Intervalle möglich, bekannte Implementierungen (etwa TSS/370, /I 1/) arbeiten jedoch mit einer Zweiklassen-Einteilung.

Überträgt man diesen Gedanken einer Working Set-Überprüfung zu Page Fault-Zeitpunkten auf das DWS-Verfahren, so stellt man fest, daß das Ergebnis das PFF-Verfahren ist, diese Überlegungen also zu einem neuen Zugang zum PFF-Verfahren führen. Wählt man nämlich in der genannten Weise das Auftreten einer Fehlseitenbe-

dingung als Überprüfungszeitpunkt, so ist es unerläßlich, daß man, um überhaupt den Aufbau eines Working Set zu ermöglichen, einen zeitlichen Mindestabstand aufeinander folgender Fehlseitenbedingungen für die Zulässigkeit der Überprüfung festlegen muß, und erhält damit exakt das PFF-Verfahren.

Neben den vorstehend beschriebenen Ansätzen zur Vereinfachung der Verfahren, um sie für die Praxis tauglich zu machen, gibt es - insbesondere beim DWS-Verfahren - auch eine Reihe von Ansätzen, die Verfahren zu verfeinern.
Beim DWS-Verfahren ist vorwiegend die nicht unkritische Wahl der Fenstergröße Ausgangspunkt weiterer Untersuchungen. T ist so zu wählen, daß weder eine zu hohe Fehlseitenrate (T zu klein) noch eine uneffiziente Arbeitsspeichernutzung (T zu groß) auftreten kann. Da das Programmverhalten aber bei verschiedenen Programmen nicht gleich sein muß und sich sogar im Verlauf eines Programmes ändern kann, liegt der Gedanke nahe, T variabel zu halten und so zu variieren, daß sich die Fehlseitenrate in vorgegebenen Schranken bewegt. Praktisch geschieht dies, indem man die Fehlseitenrate mißt und in Abhängigkeit davon T dynamisch für die nächste Zukunft festlegt.

Das Verfahren hat folgende Nachteile:

(1) Es ist nicht leicht, sinnvolle, feste obere und untere Schranken für die Fehlseitenrate festzulegen. (Dies ist im Prinzip das Parameterproblem des PFF-Verfahrens).

(2) Das DWS-Verfahren zeigt zwar reguläres Verhalten (vgl. Kap. 2.5) und der Verfahrensparameter T kann somit zur Steuerug der Fehlseitenrate herangezogen werden; es gibt jedoch - weil dies ebenfalls eine Frage des Programmverhaltens ist - keine allgemeingültige Angabe, wie stark T verändert werden muß, um eine Veränderung der Fehlseitenrate von bestimmter Größe zu erzielen; insofern ist das ursprüngliche Parameterproblem nur verschoben.

(3) Das DWS-Verfahren reagiert in negativer Weise auf Lokalitätsübergänge.

Wie bereits erläutert, führen Phasenübergänge zu einem kurzfristigen Anstieg der Fehlseitenrate, wobei in diesem Falle eine Vergrößerung von T die falsche Reaktion wäre.

Ein anderer Versuch, T den Gegebenheiten besser anzupassen, besteht darin, mit zwei festen Werten für T zu arbeiten, einen für Instruktions- und einen für Datenseiten. Die natürliche Erweiterung dieser Vorgehensweise besteht darin, für jede Programmseite die maximale unreferierte Verweildauer im Arbeitsspeicher (d.h. T(i), i = 1, ..., n) individuell festzulegen. Das Problem dieses Verfahrens (page partition algorithm /P4/ genannt) ist die Bestimmung der n Fenstergrößen.

Abschließend kann festgestellt werden, daß erhöhter Implementationsaufwand und Overhead sowie zusätzliche Parameterschwierigkeiten die meist bescheidenen erzielbaren Performance-Verbesserungen dieser Verfahren durchweg paralysieren.

Ein letzter Gedanke einer möglichen Verbesserung der bekannten Ersetzungsstrategien ist der folgende: Die Ersetzungsstrategien sollten berücksichtigen, ob eine auszutauschende Seite verändert worden ist oder nicht. Ausgangspunkt der Überlegung ist, daß ein Page Fault, bei dem die zu ersetzende Seite auf den Sekundärspeicher zurückgeschrieben werden muß, doppelt soviel kostet (nämlich zwei Sekundärspeicherzugriffe) wie wenn die zu ersetzende Seite überschrieben werden kann. Dieser Gedanke ist grundsätzlich auf alle Ersetzungsstrategien anwendbar, er bietet sich jedoch in besonderem Maße bei solchen Verfahren an, bei denen ohnedies die Kostenbalance zwischen Verweilkosten und Ladekosten eine Rolle spielen, wie es beim VMIN-Verfahren, unter bestimmtem Blickwinkel aber auch beim DWS-Verfahren, der Fall ist. Bei diesen Verfahren kann eine Berücksichtigung der Veränderung von Seiten sehr einfach durch die Bereitstellung eines zweiten, vergrößerten T-Wertes erfolgen /Y1/. Ein Vorteil ist, daß der entstehende Overhead sehr gering ist, da die entsprechenden Informationen ohnedies im System vorhanden sein müssen (Vorhandensein eines Change-Bits, das durchweg hardwaremäßig gesetzt wird).

Ein weiteres Kriterium - neben Implementationsaufwand und Performance - vor allem für die Beurteilung der vereinfachten Verfahren ist, ob und inwieweit die Regularitätseigenschaften der Basisverfahren erhalten bleiben. Dies ist Gegenstand des nachfolgenden Kapitels 2.5.

2.5 Regularitätseigenschaften

Es ist in den vorangegangenen Kapiteln schon mehrfach angeklungen, daß es aus mancherlei Gründen wünschenswert ist, wenn zwischen den Verfahrensparametern der Verfahren und den sich ergebenden Performance-Daten eindeutige, monotone Zusammenhänge bestehen. Solche Beziehungen werden nachstehend definiert und ihre Gültigkeit für reguläre Verfahren postuliert.

2.5.1 Regularität bei Verfahren fester Arbeitsspeichergröße

Bei Verfahren fester Arbeitsspeichergröße kann die vorgegebene Arbeitsspeichergröße m als Verfahrensparameter angesehen werden. Daß die Fehlseitenrate global mit fallendem m steigt, ist intuitiv einsichtig und scheint plausibel, wenn man bedenkt, daß für $m = n$ maximal n Fehlseitenbedingungen auftreten können (d. h. so viele, wie verschiedene Programmseiten vorhanden sind), während für $m = 1$ jede Referenz auf eine andere Seite zu einem Page Fault führt.

Def.: Ein Verfahren fester Speichergröße heißt regulär, wenn für alle ω gilt:

$$m < m' \Rightarrow FSR(m) \geq FSR(m'). \qquad (2.5.1)$$

Es wurde bereits sehr früh gezeigt /B4, B5/, daß z.B. das FIFO-Verfahren trotz der Richtigkeit der obigen globalen Überlegungen die Beziehung (2.5.1) nicht für beliebige m, m' erfüllt. MATTSON et al. /M2/ haben den Begriff des 'Stack Algorithmus' eingeführt und nachgewiesen, daß die Stack-Eigenschaft notwendig und hinreichend für reguläres Verhalten ist und daß eine Reihe der bekannten Seitenaustauschalgorithmen Stack-Verfahren sind. Unter einem Stack versteht man eine Permutation der Menge der Programmseiten N, $\vec{s}_t^{\,t}(\omega) = [s_1^t(\omega), s_2^t(\omega), \ldots, s_n^t(\omega)]$, so daß für alle m gilt

$$S_t(m, \omega) = \{ s_1^t(\omega), s_2^t(\omega), \ldots, s_m^t(\omega) \} \quad .$$

COFFMAN und DENNING /C6/ zeigen die Äquivalenz zwischen der Stack-Eigenschaft und der Einschlußeigenschaft

$$S_t(m, \omega) \subseteq S_t(m+1, \omega) \quad \text{für alle } m \text{ und } \omega . \tag{2.5.2}$$

Dort wird auch der Begriff des Prioritätsalgorithmus eingeführt. Ein Algorithmus A ist ein Prioritätsalgorithmus, wenn zu jeder Referenzkette $\omega = r_1 r_2 \ldots r_t \ldots$ eine Folge von Vektoren $\vec{p}_1 \vec{p}_2 \ldots \vec{p}_t \ldots$ existiert, so daß

(1) $\vec{p}_t$ eine Permutation der Seiten aus $\{\bigcup_{i=1}^{t} r_i\}$ und

(2) falls $r_{t+1} \notin S_t(m)$ und $|S_t(m)| = m$ für alle $m \geq 1$

A diejenige Seite aus $S_t(m)$ entfernt, die den höchsten Index in $\vec{p}_t$ hat.

Es wird gezeigt, daß die Existenz einer Prioritätsliste mit den obigen Eigenschaften äquivalent ist mit der Stack-Eigenschaft und der Einschlußeigenschaft.

Das LRU-Verfahren erfüllt die Einschlußeigenschaft, da S_t (LRU, m, ω) stets die m zuletzt referierten verschiedenen Seiten der Referenzkette enthält und somit eine Untermenge von S_t (LRU, m + 1, ω) ist, das die (m + 1) zuletzt referierten Seiten enthält; LRU ist somit ein reguläres Verfahren. Wegen der oben angeführten Äquivalenz existieren für das LRU-Verfahren auch ein Stack und eine Prioritätsliste; die Prioritätsliste enthält die referierten Seiten nach aufsteigender Rückwärtsdistanz geordnet.

Das LRU-Verfahren ist insofern vor allen anderen Verfahren ausgezeichnet, als es das einzige Verfahren ist, bei dem Stack und Prioritätsliste zu jedem Zeitpunkt identisch sind, und das deshalb auch als einziges Verfahren bei einer Dynamisierung des Größenparameters m die Stack-Eigenschaft nicht verliert.

Auch das OPT-Verfahren ist regulär. Die Prioritätsliste des Verfahrens ist nach aufsteigenden Vorwärtsdistanzen (und dann nach absteigenden Seitennummern) geordnet und erfüllt die geforderten Eigenschaften.

2.5.2 Regularität bei Verfahren variabler Arbeitsspeichergröße

Da bei Verfahren mit variabler Arbeitsspeichergröße sowohl die mittlere Arbeitsspeichergröße als auch die Fehlseitenrate vom Verfahrensparameter (T) abhängen, müssen sinnvollerweise zusätzlich zu einer Regularitätsbedingung zwischen mittlerer Arbeitsspeichergröße und Fehlseitenrate wie (2.5.1) entsprechende Bedingungen zwischen dem Verfahrensparameter und der Fehlseitenrate einerseits und dem Verfahrensparameter und der mittleren Arbeitsspeichergröße andererseits formuliert werden. Hinzu kommt, daß die mittlere Arbeitsspeichergröße üblicherweise aus dem Space-time Produkt berechnet wird, das (vgl. Kap. 2.1) sowohl in virtueller Zeit als auch in Realzeit gemessen werden kann, so daß die entsprechenden Beziehungen doppelt auftreten.

Im folgenden bezeichnet $\overline{m}$ die mittlere Arbeitsspeichergröße, berechnet aus dem in Prozeßzeit gemessenen Space-time Produkt und $\overline{m}_R$ die Speichergröße, berechnet aus dem in Realzeit gemessenen Space-time Produkt.

Def.: Ein Verfahren variabler Speichergröße heißt regulär, wenn für alle ω gilt

(1) die Fehlseitenrate verhält sich als Funktion der mittleren Working Set Größe monoton fallend, d.h.

$$\overline{m} < \overline{m}' \Rightarrow FSR(\overline{m}) \geq FSR(\overline{m}'), \tag{2.5.2}$$

(2) die Fehlseitenrate verhält sich als Funktion des Verfahrensparameters T monoton fallend, d.h.

$$T < T' \Rightarrow FSR(T) \geq FSR(T'), \tag{2.5.3}$$

(3) die mittlere Working Set Größe verhält sich als Funktion des Verfahrensparameters T monoton steigend, d.h.

$$T < T' \Rightarrow \overline{m}(T) \leq \overline{m}(T'). \tag{2.5.4}$$

Bei den Bedingungen (2) und (3) könnte die Monotonierichtung bei der Abhängigkeit vom Verfahrensparameter auch umgekehrt sein (gekoppelt), sie ist bei allen bekannten Verfahren jedoch wie in (2.5.3) und 2.5.4) angegeben.

Satz 2.1 Genügt der Working Set $W(t, T)$ eines Verfahrens bezüglich des Verfahrensparameters T einer Einschlußbedingung, d.h. gilt für alle ω, T, T' mit $T < T'$

$$W(t, T) \subseteq W(t, T'),$$

dann ist das Verfahren regulär.

Beweis:

(1) Da nur solche Seiten Fehlseitenbedingungen erzeugen können, die nicht im Working-Set enthalten sind, folgt aus $W(t, T) \subseteq W(t,T')$, daß bei Gültigkeit des Working Set $W(t, T')$ höchstens dann eine Fehlseitenbedingung auftreten kann, wenn auch $W(t, T)$ eine Fehlseitenbedingung liefert.

$$\Longrightarrow \quad FSR(T) \geq FSR(T'),$$

d. h. (2.5.3) gilt.

(2) Aus $W(t, T) \subseteq W(t,T')$ folgt

$|W(t, T)| \leq |W(t, T')|$ für alle t (man beachte $W(t, T) = S_t(T)$).

$$\overline{m}(T) = \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} |W(t, T)| \leq \frac{1}{|\omega|} \sum_{t=1}^{|\omega|} |W(t, T')| = \overline{m}(T'),$$

d.h. (2.5.4) ist erfüllt.

(3) Da der Working Set und damit die Working Set Größe nur von T abhängt, gilt auch die Umkehrung von (2.5.4), d. h.

$\overline{m}(T) \leq \overline{m}(T') \Longrightarrow T \leq T'$, woraus sich unter Anwendung von (2.5.3) auch 2.5.2) ergibt. q. e. d.

Wie schon vorher erwähnt, kann in (2.5.2) und 2.5.4) die mittlere Working Set Größe durch die mittlere auf Realzeit-Basis berechnete Arbeitsspeicherbelegung ersetzt werden; die analogen Bedingungen lauten dann:

$$\overline{m}_R < \overline{m}'_R \Rightarrow FSR(\overline{m}_R) \geq FSR(\overline{m}'_R) \qquad (2.5.5.)$$

und

$$T < T' \Rightarrow \overline{m}_R(T) \leq \overline{m}_R(T'). \qquad (2.5.6)$$

FRANKLIN, GRAHAM und GUPTA /F6/ zeigen an einem konkreten Gegenbeispiel beim DWS-Verfahren, daß die Gültigkeit einer Einschlußbedingung nicht hinreichend ist, um Regularität im Sinne der Bedingungen (2.5.5) und (2.5.6) zu erzwingen.

Dies wird erklärlich, wenn man sich die Formel für die mittlere Speicherbelegung in Realzeit ansieht, die sich aus der Formel (2.1.7) für das Space-time Produkt in Realzeit ergibt:

$$\overline{m}_R = \frac{\overline{m}' \cdot |\omega| + FSR \cdot |\omega| \cdot \overline{m}'' \cdot Q}{|\omega| + FSR \cdot |\omega| \cdot Q}$$

($\overline{m}'$ = mittlere Speicherbelegung in virtueller Zeit und $\overline{m}''$ = mittlere Speicherbelegung zu Page Fault Zeitpunkten).

Unterstellt man die Gültigkeit einer Einschlußbedingung, so nehmen wegen der Gültigkeit von (2.5.4) $\overline{m}'$ und $\overline{m}''$ bei Vergrößerung von T zu, so daß der erste Summand im Zähler anwächst; gleichzeitig nimmt wegen (2.5.3) die Anzahl der Fehlseitenbedingungen ab, so daß bei geeignetem ω und genügend großem Q der zweite Summand um so viel kleiner werden kann, daß der Gesamtausdruck trotz einer gleichzeitigen Verkleinerung des Nenners abnimmt.

Im folgenden sind die Begriffe 'Regularität' und 'Anomalie' stets bezogen auf die Bedingungen (2.5.2), (2.5.3) und (2.5.4) der Definition der Regularität bei Verfahren variabler Arbeitsspeichergröße und die Bedingung (2.5.1) der Definition der Regularität bei Verfahren fester Arbeitsspeichergröße, die im wesentlichen mit der Bedingung (2.5.2) identisch ist.

Das DWS-Verfahren und das VMIN-Verfahren (vgl. Kap. 2.3) genügen einer Einschlußbedingung und sind somit nach Satz 2.1 reguläre Verfahren.

Der Working Set des PFF-Verfahrens genügt als Funktion des Zeitparameters T keiner Einschlußbedingung und das Verfahren ist nicht regulär, wobei Anomalien bzgl. aller drei Regularitätsbedingungen auftreten können.

Durch welche grundsätzlichen Gegebenheiten es zu einer Verletzung der Einschlußeigenschaft kommen kann, wird anhand von Bild 2.8 erläutert.

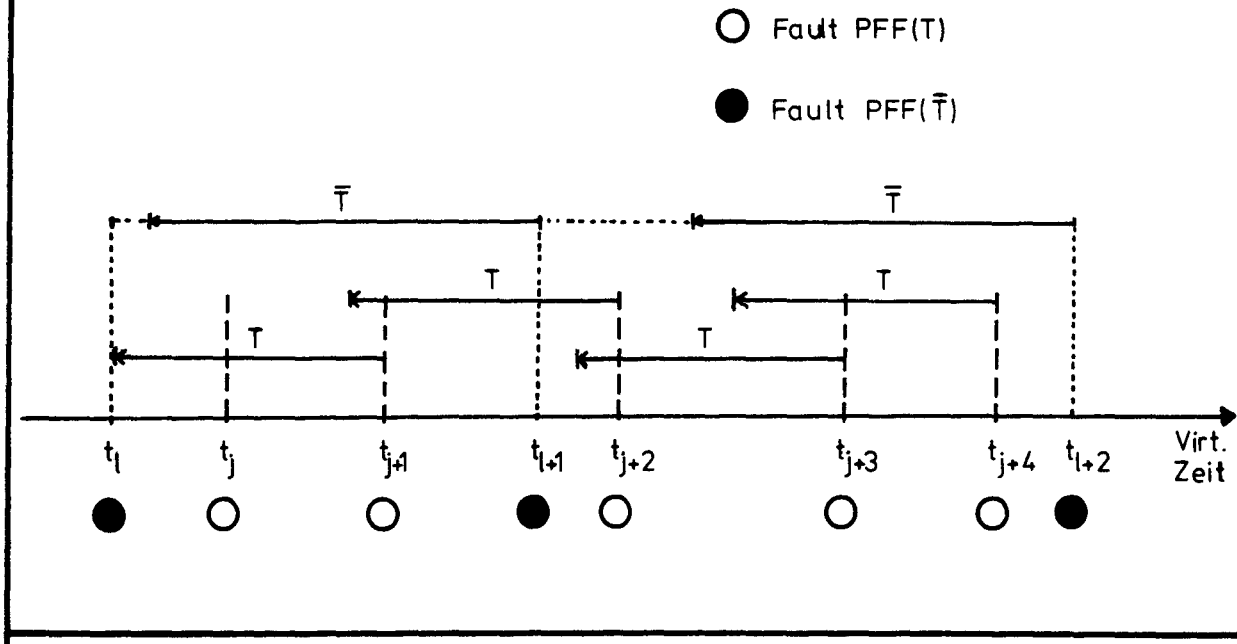


Bild 2.8
Grundsätzliche Konstellation für Ungültigkeit der Einschlußbedingung beim PFF - Verfahren

Es seien T, $\bar{T}$ mit $T < \bar{T}$ zwei Parameterwerte und t_{j+i} aufeinanderfolgende Page Fault Zeitpunkte vom PFF($\bar{T}$), und t_{l+i} aufeinanderfolgende Page Fault Zeitpunkte von PFF($\bar{T}$).

Wegen $t_{j+i} - t_{j+i-1} \leq T$ (i = 1, ..., 4) findet beim Verfahren PFF(T) keine Überprüfung des Arbeitsspeicherinhaltes zu den Zeitpunkten t_{j+i} (i = 0, ..., 4) statt und die Seiten $r_{t_{j+i}}$ (i = 0, ..., 4) werden zusätzlich geladen; wegen $t_{l+2} - t_{l+1} > \bar{T}$ findet zum Zeitpunkt t_{l+2} beim Verfahren PFF($\bar{T}$) eine Überprüfung des Arbeitsspeichers statt, bei der alle Seiten, die nach dem Zeitpunkt t_{l+1} nicht mehr referiert worden sind, aus dem Arbeitsspeicher entfernt werden. Gibt es nun Seiten, die im Abschnitt $r_{t_j} \ldots r_{t_{l+1}}$ der Referenzkette referiert worden sind und danach nicht mehr, so sind diese in $S_{t_{l+2}}(T)$ enthalten, in $S_{t_{l+2}}(\bar{T})$ jedoch nicht (dies gilt insbesondere für die Seite $r_{t_{j+1}}$, falls sie nach t_{l+1} nicht mehr referiert worden ist).

Bild 2.9 gibt ein Beispiel für eine Referenzkette, an der bei geeigneter Parameterwahl alle Anomalien demonstriert werden können.

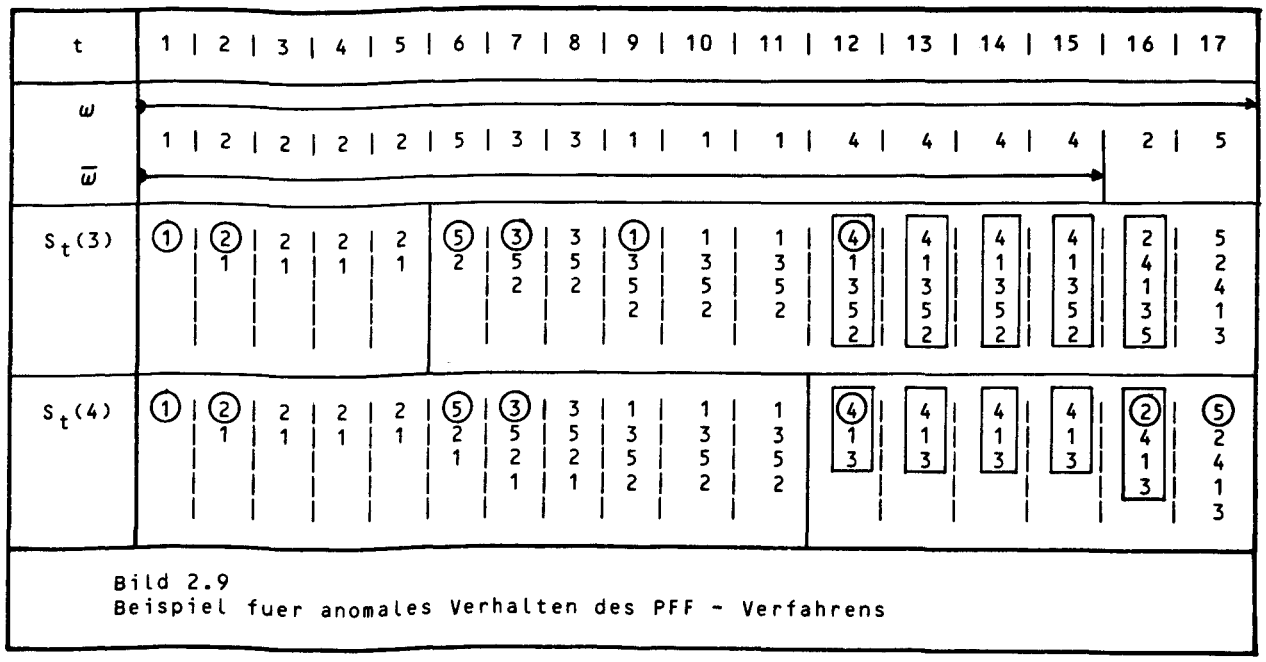


Bild 2.9
Beispiel fuer anomales Verhalten des PFF - Verfahrens

Für die Zeitpunkte t = 12 bis t = 16 ist die Einschlußeigenschaft nicht erfüllt.

Ferner ist

$$FSR(3) = \frac{6}{17} < \frac{7}{17} = FSR(4), \qquad \text{d.h. (2.5.3) ist nicht erfüllt und}$$

$$\overline{m}(3) = \frac{59}{17} > \frac{53}{17} = \overline{m}(4) \qquad \text{d.h. (2.5.4) ist nicht erfüllt.}$$

Betrachtet man als zweite Referenzkette den Abschnitt der in Bild 2.9 gegebenen bis zum Index 15, so ergibt sich für diese, $\overline{\omega}$ genannte Kette:

$$\overline{m}(3,\overline{\omega}) = \frac{49}{15} \quad \text{und} \quad FSR(3,\overline{\omega}) = 6 \quad \text{und}$$

$$\overline{m}(4,\overline{\omega}) = \frac{44}{15} \quad \text{und} \quad FSR(4,\overline{\omega}) = 5 ,$$

d.h. eine größere mittlere Arbeitsspeicherbelegung führt hier zu einer größeren Fehlseitenrate ((2.5.2) nicht erfüllt).

Die Anomalie des PFF-Verfahrens ist bei der Simulation realer Programme festgestellt worden /G3, G4/; die simulierten Laufzeiten sind jedoch nur recht kurz, so daß es nicht sicher ist, ob solche kurzfristig auftretenden Anomalien als gravierender Nachteil angesehen werden müssen.

2.5.3 Regularitätseigenschaften der vereinfachten Seitenaustauschalgorithmen

Eine interessante und in der Literatur bisher nicht untersuchte Fragestellung ist, wie weit die Regularitätseigenschaften der Originalverfahren bei den in der Praxis zum Einsatz kommenden abgeleiteten, vereinfachten Seitenaustauschverfahren erhalten bleiben.

2.5.3.1 MWS - Verfahren

Das MWS-Verfahren ist ein vom DWS-Verfahren abgeleitetes Verfahren, bei dem eine Überprüfung des Arbeitsspeicherinhaltes nach jeweils T Zeiteinheiten stattfindet.

Aus dem in Bild 2.10 angegebenen Beispiel ergibt sich

$$FSR(4, \omega) = \frac{4}{16} \text{ und } FSR(5, \omega) = \frac{5}{16},$$

d.h. eine Vergrößerung des Verfahrensparameters führt zu einer Vergrößerung der Fehlseitenrate, d.h. Parameterregularität nach (2.5.3) ist nicht gegeben;

$$\overline{m}(4, \omega) = \frac{57}{16} \text{ und } \overline{m}(5, \omega) = \frac{54}{16},$$

d. h. eine Vergrößerung des Verfahrensparameters führt zu einer Verkleinerung der mittleren Arbeitsspeicherbelegung und (2.5.4) ist nicht erfüllt.

Betrachtet man nun die verlängerte Referenzkette $\bar{\omega}$, so ergibt sich

$$\overline{m}(4, \bar{\omega}) = \frac{61}{20} \text{ und } FSR(4, \bar{\omega}) = \frac{4}{20} \text{ und}$$

$$\overline{m}(5, \omega) = \frac{62}{20} \text{ und } FSR(5, \bar{\omega}) = \frac{5}{20},$$

d. h. Widerspruch zur Regularitätsbedingung (2.5.2), wonach eine Vergrößerung der mittleren Speicherbelegung eine Verkleinerung der Fehlseitenrate nach sich zieht.

Das MWS-Verfahren zeigt also Verstöße gegen alle drei Regularitätsbedingungen und genügt überdies auch keiner Einschlußbedingung, da $S_t(4) \not\subseteq S_t(5)$ für t = 11 und t = 16 des Beispiels in Bild 2.10 ist.

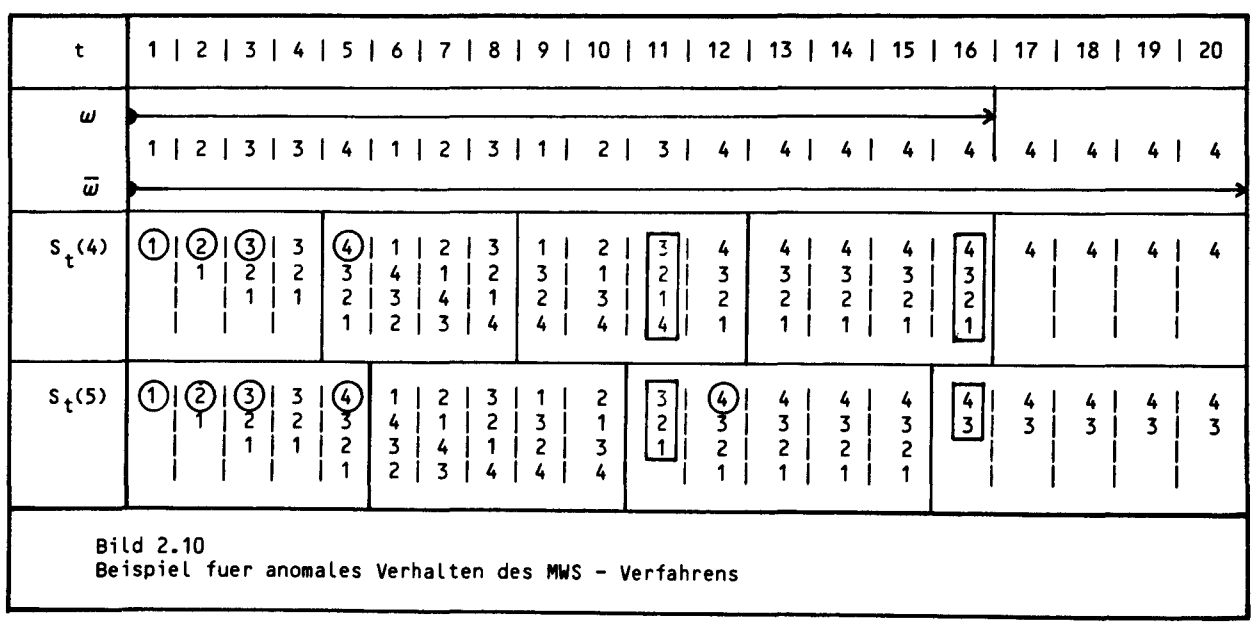


Bild 2.10
Beispiel fuer anomales Verhalten des MWS - Verfahrens

2.5.3.2 Typ-LRU-Verfahren mittels zeitgesteuerter UIC's

Wie bereits festgestellt, wird die ursprüngliche LRU-Vollordnung der Programmseiten durch eine Klasseneinteilung - wobei die Seiten mit gleichem UIC-Stand eine Klasse bilden - ersetzt. Als Kandidaten für eine Ersetzung kommen die Seiten mit höchstem UIC-Stand in Frage. Betrachtet man die Auswahl aus mehreren gleichwertigen Kandidaten der Klasse als implementations- und damit in gewissem Sinne zufallsabhängig, so gehen die Regularitätseigenschaften und wegen der Äquivalenz bei Verfahren fester Speicherzuordnung auch die Einschlußeigenschaft des LRU-Verfahrens verloren, wie das Beispiel in Bild 2.11 zeigt.

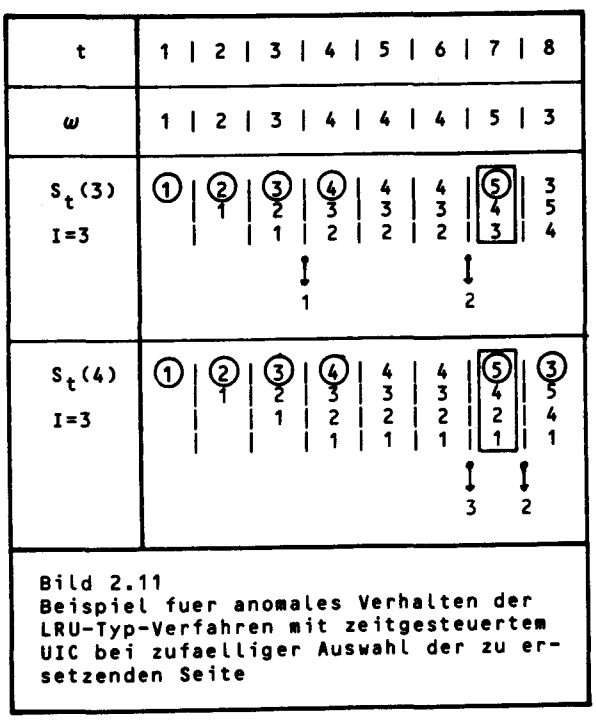


Bild 2.11
Beispiel fuer anomales Verhalten der LRU-Typ-Verfahren mit zeitgesteuertem UIC bei zufaelliger Auswahl der zu ersetzenden Seite

Für die in Bild 2.11 angegebene Referenzkette ist $FSR(3) = \frac{5}{8}$ und $FSR(4) = \frac{6}{8}$; ferner ist $S_7(3) \not\subseteq S_7(4)$.

Etabliert man zwischen den Seiten einer Klasse eine vom Größenparameter m unabhängige Ordnungsrelation, so läßt sich - wie in Satz 2.2 gezeigt wird - reguläres Verhalten der so gewonnenen Verfahren nachweisen. Eine Möglichkeit, eine solche Ordnung herzustellen, besteht darin, die Seiten einer Klasse nach der Dauer ihres unreferierten Aufenthaltes im Arbeitsspeicher zu ordnen; da die Klassen nach dem gleichen Prinzip geordnet sind (jede Seite einer bestimmten Klasse ist länger unreferiert im Speicher als irgend eine Seite einer Klasse mit niedrigerem UIC-Stand)

erkennt man unschwer, daß damit genau das LRU-Verfahren beschrieben wird und somit natürlich Regularität gegeben ist. Eine solche Ordnung der Seiten ist jedoch nicht sinnvoll, da der Sinn der Vorgehensweise ja gerade darin besteht, Verfahren zu definieren, die mit geringerem Aufwand als LRU zu realisieren sind.
Es gibt aber auch weniger aufwendige Möglichkeiten, innerhalb der Klassen eine Ordnung zu definieren, etwa nach auf- oder absteigenden Seitennummern.

Satz 2.1 Es sei A die Klasse derjenigen Verfahren, bei denen eine Klassenzuordnung nach einem zeitgesteuerten UIC vorgenommen wird und innerhalb der Klassen eine von der Arbeitsspeichergröße unabhängige Ordnungsbeziehung definiert ist; dann genügen die Verfahren der Klasse A der Einschlußeigenschaft und verhalten sich somit regulär.

Die Aussage des Satzes, die lautet

$$m < m' \Rightarrow S_t(A, m, \omega) \subseteq S_t(A, m', \omega)$$

für alle t und beliebige ω, läßt sich, wenn n die Anzahl der Programmseiten und t_A den Zeitpunkt der Referenz auf die (m + 1)-te verschiedene Seite bezeichnet, noch wie folgt präzisieren:

$$m < m' \Rightarrow \begin{cases} S_t(A, m, \omega) \equiv S_t(A, m', \omega) & \text{für} \quad t = 1, \ldots, t_A - 1 \\ S_t(A, m, \omega) \subset S_t(A, m', \omega) & \text{für} \quad t \geq t_A \ ; \end{cases}$$

ist $n \leq m$, dann gilt für alle t

$$S_t(A, m, \omega) \equiv S_t(A, m', \omega) \ .$$

Beweis: Im Verlauf des Beweises werden die Argumente A und ω weggelassen.
Die Identitätsaussagen sind trivial, da es Aussagen über die Aufbauphase des Working Set sind, die bei allen Verfahren auf identische Weise verläuft, bis zu dem Zeitpunkt, da zum ersten Mal bei einem der Verfahren eine Seite ersetzt werden muß.

Es bleibt zu beweisen, daß für alle $t \geq t_A$

$$S_t(m) \subset S_t(m')$$

ist.

Der Beweis erfolgt durch vollständige Induktion.

(a) Sei $r_{t_A} = x_A$ die zum Zeitpunkt t_A referierte Seite; da x_A die (m+1)-te verschiedene Seite in der Referenzkette ω und $m' \geq m + 1$ ist, gilt

$$S_{t_A-1}(m) \equiv S_{t_A-1}(m')$$

und

$$\begin{aligned} S_{t_A}(m) &= S_{t_A-1}(m) + \{x_A\} - \{y_A\} \\ &\subset S_{t_A-1}(m) + \{x_A\} \\ &= S_{t_A-1}(m') + \{x_A\} \\ &= S_{t_A}(m'), \end{aligned}$$

d.h. zum Zeitpunkt t_A gilt

$$S_{t_A}(m) \subset S_{t_A}(m').$$

(b) Zu zeigen: Gilt bis zu einem Zeitpunkt $t_L \geq t_A$

$$S_{t_L-1}(m) \subset S_{t_L-1}(m'),$$

so folgt

$$S_{t_L}(m) \subset S_{t_L}(m').$$

Es sind drei Fälle zu unterscheiden ($r_{t_L} = x_L$):

(1) $x_L \in S_{t_L-1}(m)$; $\Longrightarrow$ (wegen Ind.-Vorauss.) $x_L \in S_{t_L-1}(m')$.

$$\Longrightarrow S_{t_L}(m) \equiv S_{t_L-1}(m) \subset S_{t_L-1}(m') \equiv S_{t_L}(m')$$

(2) $x_L \notin S_{t_L-1}(m) \wedge x_L \in S_{t_L-1}(m')$.

$$\Longrightarrow S_{t_L}(m) = S_{t_L-1}(m) + \{x_L\} - \{y_L\}$$

$$\subset S_{t_L-1}(m) + \{x_L\}$$

$$\subseteq S_{t_L-1}(m')$$

$$= S_{t_L}(m').$$

(3) $x_L \notin S_{t_L-1}(m')$; $\Longrightarrow$ (wegen Ind.-Vorauss.) $x_L \notin S_{t_L-1}(m)$.

$$S_{t_L}(m') = S_{t_L-1}(m') + \{x_L\} - \{y'_L\}$$

$$S_{t_L}(m) = S_{t_L-1}(m') + \{x_L\} - \{y_L\}.$$

Es wird gezeigt:

$$y'_L \in S_{t_L-1}(m) \Longrightarrow y'_L = y_L .$$

Die Anordnung der für eine Ersetzung in Frage kommenden Seiten ist nach Voraussetzung vom Größenparameter m unabhängig, d. h. diejenige Seite aus $S_{t_L-1}(m')$, die dort im Sinne des UIC und der Ordnungsrelation diejenige mit niedrigster Priorität (und die zu entfernende) ist, ist es auch in $S_{t_L-1}(m)$, falls sie dort überhaupt noch enthalten ist, was wegen der Induktionsvoraussetzung

$$S_{t_L-1}(m) \subset S_{t_L-1}(m')$$

nicht notwendig der Fall ist.

$\Longrightarrow$ Es gilt entweder

$$y'_L \notin S_{t_L-1}(m),$$

$$\Longrightarrow S_{t_L-1}(m) - \{y_L\} \subset S_{t_L-1}(m)$$

$$\subseteq S_{t_L-1}(m') - \{y'_L\}$$

$$+ \{x_L\} \qquad + \{x_L\}$$

$$\Longrightarrow S_{t_L}(m) = S_{t_L-1}(m) - \{y_L\} + \{x_L\}$$

$$\subset S_{t_L-1}(m') - \{y'_L\} + \{x_L\} = S_{t_L}(m')$$

oder

$$y'_L = y_L;$$

$\Longrightarrow$ wegen $S_{t_L-1}(m) \subset S_{t_L-1}(m')$

$$S_{t_L-1}(m) - \{y_L\} \subset S_{t_L-1}(m') - \{y_L\}$$

$$+ \{x_L\} \qquad + \{x_L\}$$

$$\Longrightarrow S_{t_L}(m) = S_{t_L-1}(m) + \{x_L\} - \{y_L\}$$

$$\subset S_{t_L-1}(m') + \{x_L\} - \{y_L\} = S_{t_L}(m'),$$

d. h. es gilt in allen Fällen

$$S_{t_L}(m) \subset S_{t_L}(m').$$

Die genannten Verfahren genügen also einer Einschlußeigenschaft und sind somit regulär.

2.5.3.3 2-Klassen LRU-Verfahren auf der Basis von Fehlseitenbedingungen

Bei diesem Verfahren werden die zu ersetzenden Seiten aus der Klasse der seit dem letzten Page Fault nicht mehr referierten Seiten gewählt. Wenn diese Klasse leer ist, so ist zu wenig Speicherplatz zugeordnet (Trashing-Kondition!), und dies darf in realen Systemen nicht vorkommen bzw. bedarf einer Sonderbehandlung. (Eine ähnliche Aussage gilt für die in Kap. 2.5.3.2 behandelten Verfahren, falls alle Klassen mit UIC > 0 leer sind). Bei diesem Verfahren gilt ebenso wie bei den UIC-gesteuerten Verfahren, daß bei einer zufälligen Auswahl der zu ersetzenden Seiten die Regularitätseigenschaft des LRU-Verfahrens verloren geht, wie das Beispiel in Bild 2.12 zeigt.

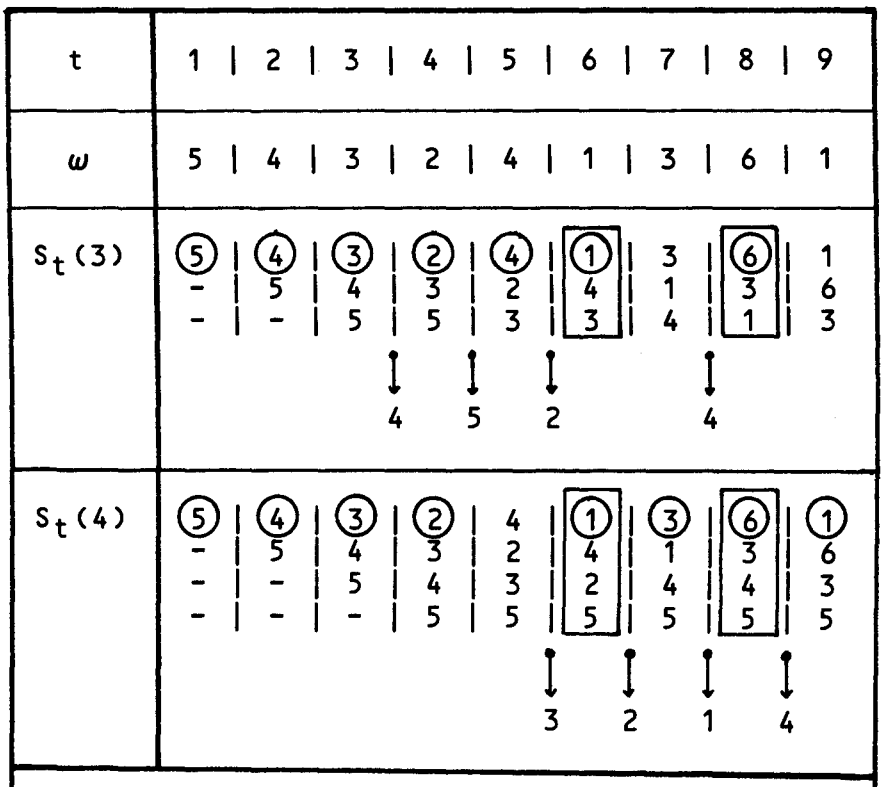


Bild 2.12
Beispiel fuer anomales Verhalten des 2-Klassen LRU-Typ-Verfahrens bei zufaelliger Auswahl der zu ersetzenden Seite

Im Beispiel in Bild 2.12 ist

$$FSR(3) = \frac{7}{9} \text{ und } FSR(4) = \frac{8}{9},$$

die Einschlußbedingung ist für t = 6 und t = 8 nicht erfüllt.

Definiert man innerhalb der Klasse der in Anspruch nehmbaren Seitenrahmen eine Ordnungsbeziehung, so sind die entstehenden Verfahren - auch wenn die Ordnungsbeziehung vom Größenparameter m unabhängig ist - im allgemeinen nicht regulär. Bild 2.13 zeigt ein Beispiel dafür, bei dem die Ordnung durch absteigende Seitennummern gegeben ist (niedrigste Seiten-Nr. = niedrigste Priorität).

t	1	2	3	4	5	6	7	8	9	10	11	12	13
ω	7	6	5	4	3	4	5	6	2	1	7	2	1
$S_t(4)$	⑦	⑥	⑤	④	③	4	⑤	⑥	②	①	7	2	1
	-	7	7	7	7	3	7	7	7	7	7	7	7
	-	-	6	6	6	7	4	5	6	6	6	1	2
	-	-	-	5	4	6	3	4	5	2	2	6	6
$S_t(5)$	⑦	⑥	⑤	④	③	4	5	6	②	①	⑦	②	①
	-	7	7	7	7	3	4	5	6	6	6	7	7
	-	-	6	6	6	7	3	4	5	5	5	6	6
	-	-	-	5	5	6	7	3	4	4	4	5	5
	-	-	-	-	4	5	6	7	3	2	1	4	2

Bild 2.13
Beispiel fuer anomales Verhalten des 2-Klassen LRU-Typ-Verfahrens bei Anordnung der Seiten nach absteigenden Seitennummern

Es ergibt sich

$$FSR(4) = \frac{9}{13} \text{ und } FSR(5) = \frac{10}{13},$$

die Einschlußbedingung ist für t = 10, 11, 12 nicht erfüllt. Die Ursache dafür liegt darin, daß der Überprüfungszeitpunkt und damit Größe und Zusammensetzung der

Seitenklassen von den Zeitpunkten des Auftretens von Fehlseitenbedingungen abhängig sind und damit vom Größenparameter m. Diese Abhängigkeit ist durch ein lokales Ordnungsprinzip nicht zu eliminieren, so daß die aufstellbaren Prioritätslisten den in Kap. 2.5.1 gegebenen Bedingungen nicht genügen.

Die Abhängigkeit vom Größenparameter m kann beseitigt werden durch globale Ordnungsprinzipien, wie etwa die Dauer des unreferierten Aufenthaltes im Arbeitsspeicher, wodurch sich wiederum das ursprüngliche LRU-Verfahren ergibt.

Abschließend kann festgestellt werden, daß die vereinfachten Seitenaustauschverfahren häufig die den Ausgangsverfahren zukommenden Regularitätseigenschaften verlieren.

2.6. Verfahren zur Simulation

Die Simulation von Seitenaustauschalgorithmen ist eine sehr aufwendige Methode zur Bestimmung der Performance; sie hat dafür aber als einzige den Vorteil, exakte Daten zu liefern, weil sie frei von Annahmen ist.

Der Aufwand kann unter Umständen jedoch durch Anwendung geschickter Methoden erheblich reduziert werden; diese bestehen vorwiegend in der Ausnutzung bestimmter Verfahrenseigenschaften, vorwiegend Stack-Eigenschaft und Einschlußeigenschaft, aber unter Umständen auch aus einer unkonventionellen Vorgehensweise, beruhend auf einer von der durch die Verfahrensdefinition nahegelegten, abweichenden Sicht der Dinge.

2.6.1 LRU - Verfahren

Ein wichtiger Vorteil der Stack-Verfahren ist, daß sich bei ihnen die Anzahl der Fehlseitenbedingungen für alle Arbeitsspeichergrößen m in sehr einfacher Weise in einem einzigen Durchlauf durch die Referenzkette ermitteln läßt.

Definiert man die Stack-Distanz $D_t(x)$ einer Seite x als die Position von x im Stack $\vec{s}_t(\omega)$ (d.h. $s_k^t(\omega) = x \Rightarrow D_t(x) = k$) und $D_t(x) = \infty$, falls die Seite x noch nicht referiert worden und somit noch nicht im Stack enthalten ist, so tritt in einem Arbeitsspeicher der Größe m eine Fehlseitenbedingung bei der Referenz der Seite x zum Zeitpunkt t genau dann auf, wenn $D_t(x) > m$ ist. Führt man nun für die Stackdistanzen 1, ..., n und ∞ n + 1 Zähler Z_1, ..., Z_n und Z_∞ (für erstmalig referierte Seiten) ein und inkrementiert bei jeder Referenz den entsprechenden Zähler, dann ist die Anzahl der Fehlseitenbedingungen

$$FS(m) = Z_\infty + \sum_{i=m+1}^{n} Z_i \quad \text{für alle} \quad m = 1, \ldots, n-1$$

und

$$FS(n) = Z_\infty .$$

Dieses Verfahren wird bei den durchgeführten LRU-Simulationen nur als Referenz-Verfahren benutzt, weil es wegen seines summarischen Charakters keine Detailinformationen über die einzelnen Page Faults (etwa Zeitpunkt, Seitennummer der verursachenden und der zu ersetzenden Seiten u.a.m.) zu liefern vermag.

Hinzu kommt, daß dieses Verfahren nicht ganz unaufwendig ist, weil für jeden virtuellen Zeitpunkt der Stack aktualisiert werden muß, eine Prozedur, die einen nicht unerheblichen, in Abhängigkeit von der Stack-Distanz wachsenden Aufwand erfordert; infolgedessen kann - insbesondere wenn die Fehlseitenrate nur für eine begrenzte Anzahl von Speichergrößen berechnet werden soll - ein Verfahren angewendet werden, für das zwar für jede vorgegebene Speichergröße das Verfahren simuliert werden muß, die einzelne Simulation jedoch erheblich unaufwendiger ist: Dazu wird bei jeder Referenz nur der Zeitpunkt (= Index) der Referenz festgehalten und nur bei Auftreten einer Fehlseitenbedingung anhand dieser Information der Stack konstruiert, was kaum aufwendiger ist als eine Stack-Aktualisierung bei einer Stack-Distanz, die so groß ist, daß ein Page Fault entsteht.

Darüber hinaus kann der Simulationsaufwand aufgrund der Gültigkeit der Einschlußbedingung im wesentlichen auf einen einzigen Durchlauf in der vorher beschriebenen Weise reduziert werden, wie nachfolgend beschrieben wird.

Sei m_o eine vorgegebene Arbeitsspeichergröße, dann gilt wegen der Einschlußeigenschaft

$$S_t(m_o) \subseteq S_t(m) \text{ für alle } m \geq m_o \text{ und alle } \omega .$$

Daraus folgt

$$\{FS(m)\} \subseteq \{FS(m_o)\} ,$$

d. h. die Menge der Fehlseitenbedingungen bei Arbeitsspeichergröße m ist eine Untermenge derjenigen von m_o, falls $m_o \leq m$ ist .

In dem zur Anwendung kommenden Verfahren wird deshalb nach der vorher beschriebenen Vorgehensweise das LRU-Verfahren einmal für ein kleinstes m_o simuliert und dabei ein File erstellt, der pro Fehlseitenbedingung folgende Informationen enthält:

(1) Nummer der Page Fault erzeugenden Seite,

(2) Zeitpunkt der Fehlseitenbedingung (= Index der Referenz),

(3) Stack - Position der die Fehlseitenbedingung erzeugenden Seite,

(4) Nummer der aus dem Arbeitsspeicher zu entfernenden Seite,

(5) Change-Bit der zu entfernenden Seite.

Aus diesen Informationen lassen sich für jedes $m > m_o$ die gleichen Informationen herleiten. Dies kann anhand der Stackposition für (1) bis (3) geschehen. Für die Angaben in (4) und (5) ist etwas mehr Aufwand erforderlich, weil hierfür zumindest Teile des Stack (zu Fault Zeitpunkten) nachgebildet werden müssen.

Kernpunkt der Überlegungen ist, daß bereits Fehlseitenraten von 10^{-3} so weit im Trashing-Bereich liegen, daß darüberliegende Werte im Grunde genommen nicht einmal von theoretischer Bedeutung sind. Man kann m_o deshalb so wählen, daß der oben beschriebene File ca. drei Größenordnungen kleiner ist als die ursprüngliche Referenzkette, so daß sich für die weitere Auswertung eine Reduktion des Aufwandes in vergleichbarer Größe ergibt.

2.6.2 OPT-Verfahren

Das OPT-Verfahren ist wie das LRU-Verfahren ein Stack-Verfahren und genügt somit der Einschlußbedingung, so daß die Simulation in der gleichen Weise wie beim LRU-Verfahren durchgeführt werden kann. Dies geschieht jedoch nicht, weil das OPT-Verfahren als Teil des PREOPT-Verfahren simuliert wird und dieses einer Einschlußbedingung nicht genügt (vgl. Kap. 3.2.1.2). Es muß deshalb die Fehlseitenrate für jede vorgegebene Speichergröße separat berechnet werden, wobei so vorgegangen wird, daß

nur zu Page Fault Zeitpunkten zusätzlicher Aufwand erforderlich ist, um die zu ersetzende Seite zu ermitteln, wozu ein Aufbau des Stack nicht notwendig ist, sondern nur die Ermittlung der Seite niedrigster Priorität der Prioritätsliste, die sich im Arbeitsspeicher befindet. Nun ist die Prioritätsliste des OPT-Verfahrens nach aufsteigenden Vorwärtsdistanzen geordnet (vgl. Kap. 2.5.1); deshalb muß bei jeder Referenz die Vorwärtsdistanz der gerade referierten Seite in eine seitenorientierte Liste eingetragen werden, so daß beim Auftreten einer Fehlseitenbedingung die Seite mit größter Vorwärtsdistanz aus dieser Liste ermittelt werden kann.

Zusätzlicher Aufwand entsteht dadurch, daß die Vorwärtsdistanzen nicht Bestandteil der Referenzdaten sind und auch während der Verarbeitung der Referenzkette zwecks Simulation des OPT-Verfahrens nicht erzeugbar sind. So muß für die Simulation des OPT- (und PREOPT-)Verfahrens ein besonderes Input-Band erstellt werden, das für jede Referenz nicht nur die Seitennummer, sondern auch die Vorwärtsdistanz enthält. Bei der Erzeugung der Vorwärtsdistanzen hilft folgende Überlegung: Die Vorwärtsdistanzen einer Referenzkette ω , die nicht ohne weiteres zu erzeugen sind, sind identisch mit den leicht erzeugbaren Rückwärtsdistanzen der Referenzkette ω^{Rev} ; d.h. zur Erstellung des OPT-Input-Bandes wird die originale Referenzkette rückwärts gelesen und die erzeugten Rückwärtsdistanzen werden der eingelesenen Information entprechend hinzugefügt.

2.6.3 DWS - Verfahren

Zur Berechnung einiger Performance-Daten für das DWS-Verfahren können zu den Stack-Distanzen des LRU-Verfahrens analoge Überlegungen angestellt werden. Beim DWS-Verfahren tritt eine Fehlseitenbedingung genau dann auf, wenn eine Seite referiert wird, deren letzte Referenz mehr als T Zeiteinheiten zurückliegt, d.h. wenn die Rückwärtsdistanz $b_t(x) > T$ (Zeitpunkt t, $r_t = x$) ist. Etabliert man für jeden auftretenden Wert von b_t einen Zähler und zählt darin die Häufigkeit des Auftretens

der betreffenden Rückwärtsdistanz , so erhält man z. B. die Anzahl der Fehlseitenbedingungen bei Fenstergröße T durch Summieren aller Zähler für Distanzen größer als T.

Nachteil des Verfahrens ist, daß nicht vorher bekannt ist, welche Distanzen (und wie viele) auftreten werden und die Zahl der möglichen Distanzen sehr groß ist ($\leq |\omega| - n + 1$).Trotzdem können unter Berücksichtigung der Häufigkeitsverteilung der Distanzen die genannten Überlegungen zu einem praktikablen Verfahren führen.

Ein solches Verfahren wurde jedoch nicht entwickelt, da aufgrund der Gültigkeit der Einschlußbedingung beim DWS-Verfahren analog zur Vorgehensweise beim LRU-Verfahren ein wenig aufwendiges Verfahren zur Simulation hergeleitet werden kann, das überdies - wie auch beim LRU-Verfahren - Detailinformationen pro Page Fault liefert.

Bevor dieses Verfahren geschildert wird, sind einige grundsätzliche Bemerkungen zur mittleren Arbeitsspeicherbelegung zu machen. Bei Verfahren mit variabler Arbeitsspeichergröße ist die Fehlseitenrate nur im Zusammenhang mit der mittleren Arbeitsspeichergröße aussagekräftig; d. h. daß bei der Simulation grundsätzlich auch die mittlere Speichergröße ermittelt werden muß, was mit Hilfe des Space-time Produktes geschehen kann. Das Space-time Produkt aber kann in realer oder virtueller Zeit berechnet werden; im letzteren Falle ist die sich ergebende Speicherbelegung mit der mittleren Working Set Größe identisch. Im folgenden sollen Eigenschaften der unterschiedlich gemessenen Speicherbelegungen verglichen werden.

Sie unterscheiden sich überhaupt nicht, falls die mittlere virtuelle Speicherbelegung gleich der mittleren Belegung zu Page Fault Zeitpunkten ist. Genau diese Annahme führt zu einer Formel für eine näherungsweise Berechnung des realen Space-time Produktes (vgl. Kap. 2.1.3). GRAHAM /G3/ hat für das DWS-Verfahren vergleichende Messungen des angenäherten und des exakten Space-time Produktes durchgeführt; das Ergebnis zeigt, daß das näherungsweise Space-time Produkt durchweg deutlich kleinere

Werte liefert als das exakt berechnete. Dies zeigt, daß die Annahme eine Unterschätzung der mittleren Speicherbelegung zu Page Fault Zeitpunkten beinhaltet. Was den qualitativen Teil der Aussage angeht, so wären dazu Messungen nicht notwendig gewesen: Unterstellt man nämlich - wie schon mehrfach geschehen - ein Phasen-/Übergangsmodell des Programmverhaltens, das auch beinhaltet, daß die Menge der Fehlseitenbedingungen an den Phasenübergängen auftritt, so zeigt das DWS-Verfahren an den Phasenübergängen die schon öfters erwähnte kurzfristige Expansion des Working Set (vgl. Bild 2.5).

Beim Übergang vom virtuellen zum realen Zeitmaß sind es demnach genau die in diesem Zeitabschnitt liegenden Working Set Größen, die mit einem von 1 auf ungefähr 10^4 (Geschwindigkeitsfaktor Haupt-/Sekundärspeicher) erhöhten Gewichtsfaktor in das reale Space-time Produkt eingehen. Umgekehrt gehen die Working Set Größen an Phasenübergängen bei virtueller Zeitmessung mit zu geringem Gewicht in die Mittelbildung ein, d. h., die Tendenz der Working Set Größendynamik an den Phasenübergängen findet bei virtueller Zeitrechnung einen geringeren Niederschlag in der mittleren Speichergröße als bei realer Zeitrechnung. Im Vergleich zu den auf der Basis realer Zeit berechneten Werte sind die auf virtueller Basis errechneten Werte der mittleren Speicherbelegung beim DWS-Verfahren, das an den Phasenübergängen durch relativ große Working Sets gekennzeichnet ist, zu klein (daher die Unterschätzung des Space-time Produktes durch Annahme der Gleichheit); umgekehrt ist die Situation beim PFF- und VMIN-Verfahren, die beide durch Einbrüche der Working Set Größe an Phasenübergangsstellen (vgl. Bild 2.6 und 2.7) gekennzeichnet sind und deren mittlere Speicherbelegung basierend auf virtueller Zeit deshalb zu groß ist. Bei virtueller Zeitmessung wird das DWS-Verfahren bei der mittleren Speicherbelegung begünstigt, PFF- und VMIN-Verfahren werden benachteiligt. Man beachte jedoch,daß sich an den qualitativen Performance-Relationen der Verfahren durch die verschiedenartige Messung der mittleren Speicherbelegung nichts ändert, da sich die Kurven der Working Set Größen (vgl. Bilder 2.5, 2.6, 2.7) an keiner Stelle schneiden.

Versucht man die alternativen mittleren Speichergrößen zu bewerten, so besitzt die auf realer Zeitmessung beruhende eine größere Nähe zur Praxis; sie erfordert dafür aber einen deutlich höheren Berechnungsaufwand (insbesondere beim DWS-und VMIN-Verfahren) und die konkrete Festlegung der Geschwindigkeitsrelation Haupt-/Sekundärspeicher und ist somit in ihren quantitativen Aussagen als Funktion dieser Vorgabe zu sehen. Die zuletzt angegebenen Gründe, zusammen mit dem schwerer wiegenden Argument, daß es ja nicht Ziel der vorliegenden Arbeit ist, die Performance der bekannten Seitenaustauschverfahren miteinander zu vergleichen, sondern jeweils die Performance-Relationen zwischen einem der bekannten Demand Paging Verfahren und davon abgeleiteten Prepaging Verfahren festzustellen, haben es gerechtfertigt erscheinen lassen, bei allen Seitenaustauschverfahren variabler Arbeitsspeichergröße, die Berechnung der mittleren Arbeitsspeichergröße auf der Basis virtueller Zeit durchzuführen. Es sollte aber abschließend darauf hingewiesen werden, daß bei allen zur Anwendung kommenden Verfahren der Simulation das Space-time Produkt in Realzeit prinzipiell berechnet werden kann, beim DWS-und VMIN-Verfahren sogar nachträglich aus den erstellten Ergebnis-Files.

Das DWS-Verfahren genügt der Einschlußeigenschaft bezüglich des Parameters T, d. h. für $T \geq T_0$ gilt

$$S_t(T_0) \subseteq S_t(T) \quad \text{für alle} \quad t \ .$$

Daraus ergibt sich, daß die Menge der Fehlseitenbedingungen, die durch DWS(T) erzeugt werden, eine Untermenge derjenigen von DWS (T_0) ist. Aufgrund dieser Tatsache läßt sich - analog zur Vorgehensweise bei Stack-Verfahren - ein File erzeugen, der für eine minimale Fenstergröße T_0 alle Fehlseiten und damit korrelierte Informationen enthält und aus dem sich die gleichen Informationen mit sehr geringem Aufwand für jedes $T > T_0$ gewinnen lassen. Dieser File enthält pro Fehlseitenbedingung die Informationen:

(1) Nummer der Page Fault erzeugenden Seite,

(2) Zeitpunkt der Fehlseitenbedingung (= Index der Referenz),

(3) Zeitpunkt der letzten zurückliegenden Referenz auf die Page Fault erzeugende Seite,

(4) Information, ob diese Seite bis zum Zeitpunkt dieser letzten Referenz (einschließlich) verändert wurde.

Sollen für ein $T > T_0$ Anzahl und Zeitpunkt der Fehlseitenbedingungen bestimmt werden, so kann dies sehr leicht geschehen:

Sei

$$DIST = (2) - (3) ,$$

d. h. die bei Fenstergröße T_0 eine Fehlseitenbedingung erzeugende Seite ist DIST Zeiteinheiten zuvor zuletzt referiert worden. Die bei T_0 zu diesem Zeitpunkt eine Fehlseitenbedingung liefernde Seite liefert genau dann auch bei Fenstergröße T eine Fehlseitenbedingung, wenn DIST > T ist.

Die in dem oben beschriebenen File enthaltenen Informationen reichen aus, um das Space-time Produkt virtuell und real zu berechnen. Die Ermittlung des virtuellen Space-time Produktes, das zur Bestimmung der mittleren Arbeitsspeicherbelegung herangezogen wird, wird nachfolgend beschrieben.

Die Formel für das virtuelle Space-time Produkt lautet

$$STP_v = \sum_{t=1}^{|\omega|} |S_t| \quad \text{mit}$$

$|S_t|$ = Anzahl der Seiten im Arbeitsspeicher zum Zeitpunkt t.

Die mittlere Arbeitsspeicherbelegung $\overline{m}$ ist gegeben durch

$$\overline{m} = \frac{STP_v}{|\omega|} .$$

Die Information, die notwendig ist, um das Space-time Produkt nach der obigen Formel zu berechnen, indem bei jeder Referenz $|S_t|$ auf STP addiert wird, kann zwar aus dem reduzierten File gewonnen werden, ist aber viel zu aufwendig, weil es letztlich wieder

auf Operationen je Referenz hinausläuft, was durch die Erzeugung dieses Files ja gerade vermieden werden sollte. Das Space-time Produkt läßt sich jedoch auch berechnen, indem nur zu Page Fault Zeitpunkten (die alle in dem reduzierten File enthalten sind) der zwischen zwei aufeinanderfolgenden Page Faults der gleichen Seite von dieser Seite gelieferte Beitrag auf STP addiert wird (vgl. Bild 2.14).

Seien $t_k(x)$ und $t_{k+1}(x)$ aufeinanderfolgende Zeitpunkte von durch Seite x erzeugten Fehlseitenbedingungen bei Fenstergröße T und $t_k' \in [t_k, t_{k+1})$ die letzte Referenz auf die Seite x vor dem Zeitpunkt t_{k+1}.

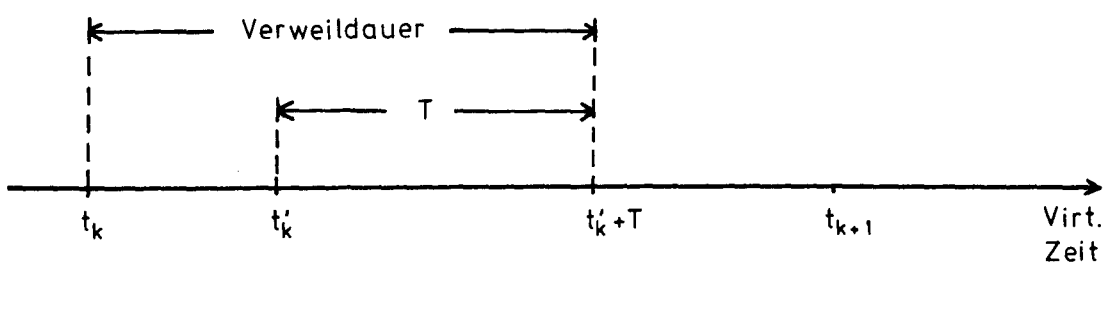


Bild 2.14
Berechnung des virtuellen STP beim DWS - Verfahren mittels Page Fault korrelierter Informationen

Die Verweildauer der Seite x im Arbeitsspeicher, nachdem sie zum Zeitpunkt t_k geladen worden ist, beträgt

$$t_k' - t_k + T + 1$$

Zeiteinheiten und ist genau der Beitrag, den die Seite x im Intervall $[t_k, t_{k+1})$ zum Space-time Produkt liefert.

Die Berechnung von $STP_v(T)$ für $T > T_0$ kann mit der Berechnung der übrigen Performance-Daten für T aus den für T_0 ermittelten Werten gekoppelt werden.

Um auch für die Randbereiche der Referenzkette die richtigen Beiträge zum Space-time Produkt bestimmen zu können, muß aus den Informationen des Files erkennbar sein, welche Fehlseitenbedingungen durch erstmalige Referenz einer Seite zustande gekommen sind (in diesem Falle gibt es keine Beitrag liefernde zurückliegende Verweilperiode der Seite), und am Ende der Referenzkette muß durch künstlich erzeugte und als solche identifizierbare Pseudo-Fehlseitenbedingungen für jede Seite der Zeitpunkt der letzten Referenz auf den File geschrieben werden. Es ist darauf zu achten, daß für Seiten, die weniger als T Zeiteinheiten vor dem Ende der Referenzkette referiert wurden, der Beitrag zum Space-time Produkt entsprechend zu korrigieren ist; er beträgt für die Seite x

$$t_{|\omega|} - t'_L(x) + 1$$

Zeiteinheiten, wenn $t_{|\omega|}$ Endzeitpunkt der Referenzkette und $t'_L(x)$ Zeitpunkt der letzten (echten) Referenz auf die Seite x und

$$t_{|\omega|} - t'_L(x) < T$$

ist.

2.6.4 VMIN - Verfahren

Die Performance-Daten für das VMIN-Verfahren werden genauso gewonnen wie für das vorher beschriebene DWS-Verfahren. Es wird dabei der gleiche für ein minimales T_0 erstellte File wie beim DWS-Verfahren benutzt und auch die Auswertung dieses Files ist nahezu gleich. Hierbei kommt zum Tragen, daß das VMIN-Verfahren mit dem DWS-Verfahren bzgl. der erzeugten Page Faults identisch ist (gleiche Fenstergröße vorausgesetzt). Die Methode, erst im nachhinein den Beitrag einer Seite zum Space-time Produkt zu bestimmen, ist beim VMIN-Verfahren unerläßlich, weil erst im nachhinein, spätestens zum Zeitpunkt t + T, festgestellt werden kann, ob die zum Zeitpunkt t referierte Seite nach diesem Zeitpunkt im Arbeitsspeicher geblieben ist oder nicht. Als einziger Unterschied zum DWS-Verfahren bleibt, daß der Beitrag zum Space-time Produkt für die vergangene Aufenthaltsperiode einer Seite im Arbeits-

speicher gegeben ist durch

$$t'_k - t_k + 1 ,$$

weil beim VMIN-Verfahren eine Seite sofort nach der letzten Referenz aus dem Arbeitsspeicher entfernt wird, falls sie überhaupt entfernt werden soll.

2.6.5 PFF - Verfahren

Das PFF-Verfahren genügt keiner Einschlußbedingung und gestattet deshalb nicht eine Vorgehensweise zur Reduktion des Simulationsaufwandes ähnlich derjenigen beim DWS-Verfahren.
Es ist bei diesen Verfahren also für jeden Parameterwert T (und ein zugehöriges T') die Verfahrenssimulation durchzuführen. Das normale PFF-Verfahren kann dabei als Grenzfall der darauf aufbauenden Prepaging-Verfahren betrachtet werden.

Bei allen Verfahren, die eine komplette Simulation pro Parameterwert erfordern - dies gilt insbesondere auch bei allen Prepaging-Verfahren (vgl. Kap. 3), da diese grundsätzlich einer Einschlußbedingung nicht genügen - werden die erforderlichen Simulationen nicht sequentiell durchgeführt, sondern parallel, so daß jede Referenz bezüglich eines Vektors von Parameterwerten verarbeitet wird. Vorteil dieser Vorgehensweise ist, daß globale, d.h. vom Parameterwert unabhängige Aufwendungen wie z. B. Zeitmessung und I/O nur einmal erbracht werden müssen.

3 PREPAGING

Gerade auch im Hinblick auf das schon mehrfach benutzte Phasen-/Übergangsmodell, das beim Aufbau einer neuen Lokalität an den Phasenübergangsstellen von einer erhöhten Fehlseitenrate ausgeht, ist der Gedanke attraktiv, die zur neuen Lokalität gehörenden Seiten auf einmal (d. h. durch Prepaging) zu laden, anstatt in einer Folge kurz aufeinanderfolgender Page Faults. Ein großes Handicap des Prepaging besteht darin, daß in der Praxis mit vertretbarem Aufwand nur schwer zuverlässige Informationen darüber zu erlangen sind, welche Seiten aufgrund des zukünftigen Programmverhaltens sinnvollerweise bereits vorab (d. h. bevor sie referiert werden) in den Arbeitsspeicher geladen werden sollten. So besteht bei realisierbaren Prepaging Verfahren grundsätzlich das Risiko, durch das Laden nicht benötigter Seiten die Anzahl der Ladevorgänge gegenüber Demand Verfahren deutlich zu erhöhen und dabei auch noch die Performance (z. B. Fehlseitenrate) zu verschlechtern oder zumindest nicht zu verbessern.

Bei dem für optimale Verfahren notwendigen Kenntnisstand (nämlich Vorabkenntnisse des Programmverhaltens, manifestiert durch Kenntnis der gesamten Referenzkette) besteht diese Gefahr nicht, und Demand Paging ist unter solchen Voraussetzungen eigentlich eine unnötige Einschränkung. Man kann also erwarten, daß sich gerade bei optimalen Verfahren durch Prepaging deutliche Performance-Verbesserungen gegenüber Demand Paging erzielen lassen.

Zunächst jedoch sind für Prepaging Verfahren geeignete Performance-Maße bereitzustellen.

3.1. Performance - Maße

Die Auswirkungen des Prepaging und das Vermögen, die Performance von Prepaging Verfahren angemessen zu beschreiben, werden an zweien der bekannten Performance-Maße untersucht.

3.1.1 Space-time Produkt

Die Definition des Space-time Produktes in virtueller Zeit ist identisch mit der in (2.1.5) gegebenen für Demand Paging Verfahren:

$$STP_V = \sum_{t=1}^{|\omega|} |S_t| \ .$$

Zu beachten ist, daß in der Beziehung

$$S_t = S_{t-1} + X_t - Y_t$$

die Menge X_t nicht mehr notwendig maximal einelementig ist, so daß die Ungleichung

$$|S_t| \leq |S_{t-1}| + 1$$

nicht mehr gültig ist.

Bei der Definition des Space-time Produktes in Realzeit sind Ergänzungen erforderlich:

$$STP_R = STP_V + \sum_{t=1}^{|\omega|} Q \cdot |S_t| \cdot \triangle(t) \cdot l(t) \cdot g(l(t))$$

$$= \sum_{t=1}^{|\omega|} |S_t| \cdot (1 + Q \cdot \triangle(t) \cdot l(t) \cdot g(l(t)))$$

mit

$$\triangle(t) = \begin{cases} 1, \text{ falls t Zeitpunkt einer Fehlseitenbedingung ist,} \\ 0 \quad \text{sonst,} \end{cases}$$

$l(t)$ = Anzahl der zum Zeitpunkt t zu ladenden Seiten,

$g(j)$ = Korrekturfunktion, um dem Tatbestand Rechnung zu tragen, daß einmaliges Laden von j Seiten weniger Aufwand beinhaltet als das j-malige Laden einer Seite ($\Longrightarrow g(j) \leq 1$, $j \cdot g(j)$ konkav).

Betrachtet man das virtuelle Space-time Produkt, so zeigt es sich, daß bei gleichbleibender Ersetzungsstrategie durch Übergang von einer Demand Ladestrategie auf eine Prepaging Strategie das Space-time Produkt nur vergrößert werden kann; dies liegt daran, daß es - auch bei optimalen Prepaging Verfahren - typisch ist, daß Seiten bereits vor dem Referenzzeitpunkt in den Arbeitsspeicher geladen werden und somit bis zum Zeitpunkt ihrer Referenz einen Beitrag zum Space-time Produkt liefern, der bei einer Demand Ladestrategie nicht anfallen würde.

Da STP_V als Summand auch im in Realzeit gemessenen Space-time Produkt enthalten ist, tritt auch dort zunächst eine Vergrößerung auf, die nur durch eine entsprechende Verkleinerung des zweiten Terms aufgewogen werden könnte. Zumindest bei optimalen Verfahren könnte dies bei einer entsprechenden Begünstigung größerer Übertragungseinheiten durch die Funktion g(j) geschehen. Bei nichtoptimalen Verfahren, bei denen davon ausgegangen werden muß, daß die Zahl der zu ladenden Seiten größer ist, als bei dem zum Vergleich stehenden Demand Verfahren, dürfte der durch g(j) charakterisierte Gewinn jedoch kaum ausreichen, um den negativen Einfluß einer größeren Anzahl zu ladender Seiten und des virtuellen Space-time Produktes auszugleichen.

Zusammenfassend muß gesagt werden, daß beim Performance-Maß Space-time Produkt Prepaging Verfahren gegenüber Demand Paging Verfahren benachteiligt werden. Die Ursache liegt auf der Hand: Das Space-time Produkt ist bestimmt durch die mittlere Arbeitsspeicherbelegung und die Anzahl der Ladevorgänge (weniger durch die Anzahl der Fehlseitenbedingungen; diese Unterscheidung ist bei Demand Paging wegen der

Übereinstimmung beider Werte nicht relevant), d.h., die verfahrenstypischen Nachteile - nämlich im Mittel eine höhere Speicherbelegung und vergrößerte Zahl von Ladevorgängen - werden voll erfaßt, während die potentiellen Vorteile - nämlich eine Reduktion der Fehlseitenrate - fast keinen Niederschlag findet.

Eine Berücksichtigung der Fehlseitenrate könnte z. B. durch eine Berechnung des Space-time Produktes in Systemzeit erreicht werden, bei der auch die außerhalb des Programms liegenden Verlustzeiten berücksichtigt werden. Global kann man sich diese Zeiten (Kosten) auch durch Q und g(j) berücksichtigt denken; dies soll hier nicht geschehen, da die pro Fehlseitenbedingung auftretenden Kosten genauer aufgeschlüsselt werden sollen.

$$\hat{Q} = A + B + C$$

mit

A = Systemoverhead (Vorbereitung des Page I/O, Programmwechsel u. ä.).

B = $Q \cdot l(t) \cdot g(l(t))$ = durch Page I/O verursachte Warte-und Transferzeiten.

C = Wartezeit des wieder rechenbereiten Programms bis die Ausführung fortgesetzt wird.

Der Beitrag B ist im wesentlichen, d.h. abgesehen von einer nicht allzu gravierenden Korrektur durch g(j) proportional zur Anzahl der zu ladenden Seiten; A und C sind nahezu unabhängig von der Anzahl der zu ladenden Seiten und fallen einmal pro Page Fault an.

Nun sprechen viele Anzeichen dafür, daß in der Praxis bei unter voller Last laufenden Systemen A und insbesondere C einen erheblichen Beitrag liefern; hinzu kommt, daß bei schnelleren Sekundärspeichern, wie sie für die Zukunft denkbar sind, der relative Anteil der Summanden A und C an $\hat{Q}$ entsprechend ansteigen würde.

Leider sind die Größen A und C dynamisch variabel, so daß ein darauf aufbauendes Space-time Produkt umgebungsabhängig nicht reproduzierbare und damit für Vergleiche unbrauchbare Performance-Daten liefern würde. Unter der Voraussetzung eines stationären Lastzustandes jedoch wären diese Größen zwar installationstypisch, aber fest. Es gibt jedoch keine Messungen, die eine Quantifizierung dieser Größen gestatten würden, so daß das Space-time Produkt in Systemzeit nicht sinnvoll verwendet werden kann, obwohl es grundsätzlich geeignet ist, Vorteile und Nachteile von Prepaging Verfahren voll zu erfassen und somit für Vergleiche zwischen Demand - und Prepaging Verfahren geeignet wäre.

3.1.2 Fehlseitenrate

Wenn es auch keineswegs sicher ist, daß durch Einführung einer Prepaging Ladestrategie die Anzahl der Fehlseitenbedingungen reduziert werden kann, so ist das doch derjenige Effekt, den man - zumindest bei einem funktionierenden Verfahren - erwarten kann, und dieser wird durch die Fehlseitenrate voll erfaßt. Die Fehlseitenrate ist also geeignet, die potentiellen Vorteile des Prepaging zu erfassen. Auch die bei Verfahren variabler Arbeitsspeichergröße i. a. größere mittlere Arbeitsspeicherbelegung wird erfaßt, da $\overline{m}$ Bestimmungsparameter der Fehlseitenrate ist und überlicherweise die Fehlseitenrate über der mittleren Arbeitsspeicherbelegung aufgetragen wird.

Nicht erfaßt wird durch die Fehlseitenrate die gegenüber Demand Verfahren im allgemeinen erhöhte Anzahl der Ladevorgänge, so daß über alles gesehen dieses Performance-Maß Prepaging Verfahren begünstigt.

3.1.3 Modifizierte Fehlseitenrate (MFSR)

Um auch die erhöhte Anzahl von Ladevorgängen erfassen zu können, wird eine Modifikation der Fehlseitenrate vorgeschlagen:

$$MFSR(A, \bar{m}, \omega) = \frac{1}{|\omega|}\sqrt{FS(A, \bar{m}, \omega) \cdot X(A, \bar{m}, \omega)}, \qquad (3.1.1)$$

wobei X die Gesamtzahl der Ladevorgänge bezeichnet.

Für Demand Paging Verfahren gilt wegen X = FS

$$MFSR \equiv FSR .$$

Die in (3.1.1) definierte modifizierte Fehlseitenrate kann auch als Sonderfall einer allgemeineren Definition aufgefaßt werden, die eine unterschiedliche Gewichtung der beteiligten Größen (etwa in Abhängigkeit des installationstypischen Belastungsprofils der Systemkomponenten) zuläßt. Da hier aber das Problem einer sinnvollen Wichtung nicht diskutiert werden kann, wird von vornherein gleiches Gewicht für beide Größen unterstellt.

Wegen der engen Verknüpfung zwischen Fehlseitenrate und mittlerer Gültigkeitsdauer gelten die Aussagen in Kapitel 3.1.2 und 3.1.3 in analoger Weise auch für dieses Performance-Maß.

3.2. Optimale Prepaging Verfahren

3.2.1 Verfahren fester Arbeitsspeichergröße

Es werden in diesem Abschnitt zunächst zwei Sätze bewiesen, die das Verhältnis zwischen Demand Paging Verfahren und Prepaging Verfahren bezüglich bestimmter Optimalitätskriterien beleuchten. Danach werden zwei nach verschiedenen Kriterien optimale Prepaging Verfahren vorgestellt.

3.2.1.1 Allgemeines

Wie vorher ist die Menge der zur Zeit t im Arbeitsspeicher befindlichen Seiten gegeben durch

$$S_t = S_{t-1} + X_t - Y_t$$

für alle $t > 0$ und $S_0 = \emptyset$. Alle Größen sind vom Austauschalgorithmus A, vom Programm ω und von der Arbeitsspeichergröße m abhängig; explizit wird die Abhängigkeit im folgenden nur dort angegeben, wo es für die Aussage von Bedeutung ist. Ferner sei $X = X(A, \omega, m)$ die Anzahl der durch das Verfahren A bei der Abarbeitung von ω in einem Arbeitsspeicher der Größe m zu ladenden Seiten; dann ist

$$X = \sum_{t=1}^{|\omega|} |X_t| .$$

Satz 3.1 Zu jedem Paging Verfahren A existiert ein Demand Paging Verfahren A', so daß

$$X(A') \leq X(A)$$

ist für alle ω und m.

Beweis: Der Beweis erfolgt durch Konstruktion eines Algorithmus A' mit der geforderten Eigenschaft. Die Aussage ist trivial, falls A selbst ein Demand

Verfahren ist, weil A' = A die Aussage erfüllt; es wird deshalb A als Prepaging Algorithmus vorausgesetzt.

Der Algorithmus A' wird so gebildet, daß er den Algorithmus A nachempfindet, indem über die zu jedem Zeitpunkt t von A durchgeführten Aktionen Buch geführt wird und von diesen Aktionen diejenigen nachvollzogen werden, die durch das Auftreten von Fehlseitenbedingungen erzwungen werden

Der Algorithmus A erzeugt die Arbeitsspeicherzustandsfolge $\{S_t\}$ mit

$$S_t = S_{t-1} + X_t - Y_t \text{ mit } X_t \cap Y_t = \emptyset .$$

Analog dazu erzeugt der Algorithmus A' die Folge $\{S'_t\}$ mit

$$S'_t = S'_{t-1} + X'_t - Y'_t ,$$

wobei

$$X'_t = \begin{cases} \emptyset \text{, falls } r_t \in S'_{t-1} \\ \{r_t\} \text{ sonst} \end{cases} \qquad (3.2.1)$$

und

$$Y'_t = \begin{cases} \emptyset \text{, falls } X'_t = \emptyset \text{ oder } |S'_{t-1}| < m \\ \{y_t\} \text{ sonst .} \end{cases} \qquad (3.2.2)$$

Sei P_t die Menge der zum Zeitpunkt t von A aber nicht von A' geladenen Seiten (zurückgestellte Ladevorgänge) und R_t die Menge der zum Zeitpunkt t von A aber nicht von A' aus dem Arbeitsspeicher entfernten Seiten (zurückgestellte Ersetzungsvorgänge).

Dann gilt

$$S'_t = S_t + R_t - P_t$$

und

$$R_t = (R_{t-1} + Y_t - X_t) \cap S'_t ,$$

d.h. eine Seite ist genau dann zum Zeitpunkt t eine zurückgestellte Ersetzung, wenn sie es bereits zum Zeitpunkt t - 1 war und nicht zum Zeitpunkt t durch A geladen wurde, oder wenn sie zum Zeitpunkt t durch A ersetzt wurde, aber in S'_t enthalten ist. Somit gilt für das Element y_t in (3.2.2)

$$y_t \in \begin{cases} R_{t-1}, \text{ falls } R_{t-1} \neq \emptyset \\ Y_t \text{ sonst.} \end{cases}$$

Falls R_{t-1} od. alternativ Y_t mehrelementig sind, ist das Verfahren A' nicht eindeutig, es kann jedoch durch eine geeignete Auswahlvorschrift für y_t eindeutig gemacht werden.

Für die Menge der zurückgestellten Ladevorgänge P_t gilt:

$$P_t = (P_{t-1} + X_t - Y_t) - (S'_{t-1} \cup \{r_t\}) ,$$

d. h., eine Seite ist genau dann zum Zeitpunkt t ein zurückgestellter Ladevorgang, wenn sie es entweder schon zum Zeitpunkt t - 1 war und durch das Verfahren A nicht zum Zeitpunkt t aus dem Arbeitsspeicher entfernt wurde, oder zum Zeitpunkt t durch A geladen wurde, aber nicht bereits in S'_{t-1} enthalten war und auch nicht durch A' zum Zeitpunkt t geladen wurde. Somit gilt für das Element x in (3.2.1)

$$x \in P_{t-1}, \text{ falls } x \notin X_t .$$

Aus der Konstruktion des Verfahrens A' ergibt sich somit, daß - wann immer A' eine Fehlseitenbedingung erzeugt (also durch A' eine Seite geladen wird) - entweder durch A ebenfalls eine Fehlseitenbedingung erzeugt wird (und mindestens eine Seite geladen wird) oder die Fehlseiten-

bedingung verursachende Seite ein zurückgestellter Ladevorgang ist und somit von A bereits zu einem früheren Zeitpunkt geladen worden ist.

$\Longrightarrow$ Für jedes t gilt

$$\sum_{i=1}^{t} |X'_t| \leq \sum_{i=1}^{t} |X_t|$$

$\Longrightarrow$ $X(A', \omega, m) \leq X(A, \omega, m)$. q.e.d.

Mit Hilfe des Satzes 3.1. läßt sich nun ein Satz beweisen, der die Optimalität des OPT-Verfahrens bezüglich der Anzahl der Ladevorgänge über die Klasse der Demand Verfahren hinaus zeigt.

Satz 3.2 Der Seitenaustauschalgorithmus OPT ist bezüglich der Anzahl der zu ladenden Seiten optimal, d.h. es gilt

$$X(OPT, \omega, m) \leq X(A, \omega, m)$$

für beliebige Algorithmen A mit fester Arbeitsspeichergröße.

Beweis: Bezüglich des Beweises der Optimalität des OPT-Verfahrens in der Klasse der Demand Verfahren wird auf /M 2/ oder /C 6/ verwiesen.

Die Aussage des Satzes ist trivial für alle Demand Paging Verfahren A, da OPT bei Demand Paging Verfahren bzgl. der Anzahl der erzeugten Fehlseitenbedingungen optimal ist und diese bei Demand Verfahren mit der Anzahl der geladenen Seiten übereinstimmt.

$\Longrightarrow$ A sei ein Prepaging Algorithmus.

Beweis indirekt:

Sei A ein Prepaging Algorithmus, für den gelte

$$X(OPT, m, \omega) \geq X(A, m, \omega). \qquad (3.2.3)$$

Nach Satz 3.1 gibt es zu A einen Demand Paging Algorithmus A', so daß

$$X(A, m, \omega) \geq X(A', m, \omega)$$

gilt. Zusammen mit (3.2.3) folgt daraus

$$X(OPT) \geq X(A) \geq X(A'),$$

wobei OPT und A' Demand Paging Algorithmen sind. Dies ist ein Widerspruch zur Optimalität von OPT in der Kasse der Demand Paging Verfahren.

$\Longrightarrow$ Annahme (3.2.3) ist falsch, und es gilt stets

$X(OPT, m, \omega) \leq X(A, m, \omega)$ für alle m, ω, A. q.e.d.

Im folgenden werden zwei in verschiedener Hinsicht optimale Prepaging Verfahren definiert. Das eine Verfahren, PREOPT genannt, basiert auf dem OPT-Verfahren und liefert die gleiche - also minimale - Anzahl Ladevorgänge; das andere Verfahren, OPR, ist bezüglich der Anzahl der erzeugten Fehlseitenbedingungen optimal, liefert i. a. aber eine nicht minimale Anzahl von Ladevorgängen.

3.2.1.2 Das PREOPT-Verfahren

Das OPT-Verfahren ist dadurch definiert, daß seine Prioritätsliste nach aufsteigenden Vorwärtsdistanzen geordnet ist, wobei, falls zu einem Zeitpunkt t für mehrere x_i $d_t(x_i) = \infty$ gilt, durch eine besondere Ordnungsregel (etwa lexikographische Anordnung der Seitennummern) die Eindeutigkeit des Verfahrens erzielt wird. Diese Regelung soll - ohne explizit erwähnt zu werden - bei dem im folgenden definierten PREOPT-Verfahren in gleicher Weise wie bei dem zugrunde liegenden OPT-Verfahren gelten.

Das PREOPT-Verfahren besteht darin, daß beim Auftreten einer Fehlseitenbedingung versucht wird, mehrere Seiten, die beim OPT-Verfahren durch aufeinanderfolgende Fehlseitenbedingungen geladen würden, auf einmal zu laden, und zwar so, daß die Arbeitsspeicherinhalte beider Verfahren zu den Zeitpunkten gemeinsamer Fehlseitenbedingungen übereinstimmen.

Es sei

$L_t^{t'}(\omega)$ die Menge der verschiedenen Seiten im Abschnitt $r_t \ldots r_{t'}$ der Referenzkette ω, d.h.

$$L_t^{t'} = \{ x_i \mid x_i \in \{r_t \ldots r_{t'}\}, x_i \neq x_m \text{ für } i \neq m \},$$

$K_t^{t'}(A, m, \omega)$ die Menge der verschiedenen Seiten des Abschnitts $r_t \ldots r_{t'}$ der Referenzkette ω, die nicht in S_{t-1} enthalten sind (d. h. $K_t^{t'} = L_t^{t'} - S_{t-1}$) und

$R_t^{t'}(A, m, \omega)$ die Menge der Seiten, die durch das Verfahren A im Abschnitt $r_t \ldots r_{t'}$ aus dem Arbeitsspeicher entfernt werden, d. h.

$$R_t^{t'} \subseteq S_{t-1} \cup L_t^{t'}.$$

Man beachte die Identität $S_{t-1} \cup L_t^{t'} \equiv S_{t-1} \cup K_t^{t'}$.

Es sei t_j Zeitpunkt einer gemeinsamen Fehlseitenbedingung ($t_j = 1$ ist der erste) und $t_{j+1}, t_{j+2}, \ldots$ die Zeitpunkte der nachfolgenden Fehlseitenbedingungen des OPT-Verfahrens; $|S_{t_j-1}| = m$ vorausgesetzt, ist

$$y_{t_j} = \max_{u \in S_{t_j-1}} [d_{t_j}(u)]$$

diejenige Seite, die durch das OPT-Verfahren zum Zeitpunkt t_j aus dem Arbeitsspeicher entfernt wird,

$$y_{t_{j+1}} = \max_{u \in S_{t_j-1} \cup \{r_{t_j}\} - \{y_{t_j}\}} [d_{t_{j+1}}(u)]$$

diejenige Seite, die zum Zeitpunkt t_{j+1} entfernt wird

.
.
.

$$y_{t_{j+i}} = \max_{u \in S_{t_{j-1}} \cup \{r_{t_j}, r_{t_{j+1}}, \ldots r_{t_{j+i-1}}\} - \{y_{t_j}, y_{t_{j+1}}, \ldots, y_{t_{j+i-1}}\}} [d_{t_{j+i}}(u)]$$

diejenige Seite, die zum Zeitpunkt t_{j+i} entfernt wird.

Mit den vorher eingeführten Bezeichnungen läßt sich $y_{t_{j+i}}$ wie folgt schreiben

$$y_{t_{j+i}} = \max_{u \in S_{t_j-1} \cup L_{t_j}^{t_{j+i}} - R_{t_j}^{t_{j+i}}} [d_{t_{j+i}}(u)]$$

Sei nun

$$i_j = \max_i (i \mid y_{t_{j+i}} \notin L_{t_j}^{t_{j+i}}), \tag{3.2.4}$$

dann wird das PREOPT-Verfahren wie folgt definiert ($r_{t_j} \notin S_{t_j-1}$):

$$X_{t_j} = K_{t_j}^{t_{j+i_j}}(OPT, m, \omega)$$

$$Y_{t_j} = R_{t_j}^{t_{j+i_j}}(OPT, m, \omega) \text{ für } t_j > 1 \tag{3.2.5}$$

$$Y_1 = \emptyset$$

mit $R_{t_j}^{t_{j+i_j}} = \{y_{t_j} \ldots y_{t_{j+i_j}}\}$.

Man beachte, daß das PREOPT-Verfahren eine Aufbauphase im eigentlichen Sinne nicht kennt, weil beim Programmstart (d. h. bei der ersten Fehlseitenbedingung) die ersten m verschiedenen Seiten einer Referenzkette auf einmal geladen werden.

Satz 3.3 Das Prepaging Verfahren PREOPT hat folgende Eigenschaften:

(1) Die Anzahl der bei der Verarbeitung einer Referenzkette geladenen Seiten ist minimal, d. h.

$$X(\text{PREOPT}, m, \omega) = X(\text{OPT}, m, \omega), \tag{3.2.6}$$

(2) die durch PREOPT erzeugte Fehlseitenrate ist kleiner als die des OPT-Verfahrens, d. h.

$$FSR(\text{PREOPT}) \leq FSR(\text{OPT}). \tag{3.2.7}$$

Beweis: Beide Aussagen folgen aus der Konstruktion von PREOPT, das das OPT-Verfahren nachempfindet und den Prepaging Vorgang abbricht, sobald sich eine Differenz zum OPT-Verfahren ergeben würde.

Zu (2) Beh.: Jede Referenz einer Seite, die beim PREOPT-Verfahren eine Fehlseitenbedingung erzeugt, erzeugt auch eine Fehlseitenbedingung beim OPT-Verfahren.

Diese Aussage ist richtig für $t = 1$. Sei t_j Zeitpunkt einer gemeinsamen Fehlseitenbedingung. Zum Zeitpunkt t_j wird bei beiden Verfahren die diese Bedingung verursachende Seite r_{t_j} geladen; beim PREOPT-Verfahren können noch weitere Seiten - nämlich solche, die beim OPT-Verfahren die nachfolgenden Page Faults verursachen - durch Prepaging geladen werden. Um dies bei vorgegebener Arbeitsspreichergröße tun zu können, müssen, abgesehen von der Aufbauphase, aus dem derzeitigen Arbeitsspeicher-

inhalt (S_{t_j-1}) vorzeitig Seiten entfernt werden, die beim OPT-Verfahren erst beim Auftreten der Fehlseitenbedingungen entfernt würden. Wegen der Abbruchbedingung (3.2.4) für das Prepaging, die sicherstellt, daß nur solche Seiten vorzeitig entladen werden, die in dem durch Prepaging überdeckten Abschnitt der Referenzkette nicht referiert werden, ist sichergestellt, daß durch diese vorzeitigen Entladevorgänge keine Fehlseitenbedingungen induziert werden. Daraus folgt, daß die nächstfolgende Fehlseitenbedingung des PREOPT-Verfahrens auch wieder Fehlseitenbedingung des OPT-Verfahrens ist und daß zu diesem Zeitpunkt die Arbeitsspeicherinhalte wieder übereinstimmen.

Im Grenzfall ist das PREOPT-Verfahren mit dem OPT-Verfahren identisch. Dieser Grenzfall tritt jedoch nur ein, wenn durch $m = 1$ jedes Prepaging verhindert wird. Für $m \geq 2$ liefert das PREOPT-Verfahren auf jeden Fall eine niedrigere Fehlseitenrate, weil zumindest zum Zeitpunkt $t = 1$ m Seiten auf einmal geladen werden. Die Aussage 3.1.7 läßt sich also wie folgt verschärfen:

$FSR(PREOPT, 1, \omega) = FSR(OPT, 1, \omega)$ und
$FSR(PREOPT, m, \omega) < FSR(OPT, m, \omega)$ für $m \geq 2$ und $n > 1$.

Zu (1) Da das PREOPT-Verfahren Fehlseitenbedingungen nur gemeinsam mit dem OPT-Verfahren hat und zu diesen Zeitpunkten aber nur solche Seiten durch Prepaging geladen werden, die beim OPT-Verfahren nachfolgende Fehlseitenbedingungen erzeugen und also auch geladen werden müssen, ist die Aussage richtig.

Das PREOPT-Verfahren basiert darauf, durch möglichst weitgehendes Prepaging die Abstände zwischen aufeinanderfolgenden Fehlseitenbedingungen zu maximieren;

limitierende Bedingung für das Prepaging ist dabei, daß keine grundlegende Differenz zum unterliegenden OPT-Verfahren entstehen darf. Es ist jedoch nicht à priori klar, daß durch maximales Prepaging an jeder Stelle auch das global beste auf dem OPT-Verfahren basierende Prepaging Verfahren entsteht. Eine Aussage darüber macht der nachfolgende Satz 3.4.

Satz 3.4 Sei t_k der Index der k-ten Fehlseitenbedingung des PREOPT-Verfahrens und t'_k der Index der k-ten Fehlseitenbedingung eines ebenfalls auf der Basis des OPT-Verfahrens definierten Prepaging Verfahrens. Gilt dann bei gleicher Arbeitsspeichergröße m an irgendeiner Stelle

$$t'_k \leq t_k ,$$

dann gilt auch

$$t'_{k+1} \leq t_{k+1} .$$

Der Satz besagt, daß durch Verzicht auf maximales Prepaging ($t'_k \leq t_k$) an irgendeiner Stelle die Fehlseitenrate des Verfahrens nicht verbessert werden kann.

Beweis: Seien t_l die Zeitpunkte der Fehlseitenbedingungen des beiden Prepaging Verfahren gemeinsamen Basisverfahrens OPT(m) und bezeichne $\{FS(A, m)\}$ die Menge der Zeitpunkte, an denen das Verfahren A Fehlseitenbedingungen liefert, wenn ein Areitsspeicher der Größe m zur Verfügung steht (z. B. $\{FS(OPT, m)\} = \{t_l \mid t_l \leq |\omega|\}$).

Da die Menge der Fehlseitenbedingungen des OPT-Verfahrens eine Übermenge der Fehlseitenbedingungen der Prepaging Verfahren ist, gibt es Indizes l und l' des OPT-Verfahrens, so daß

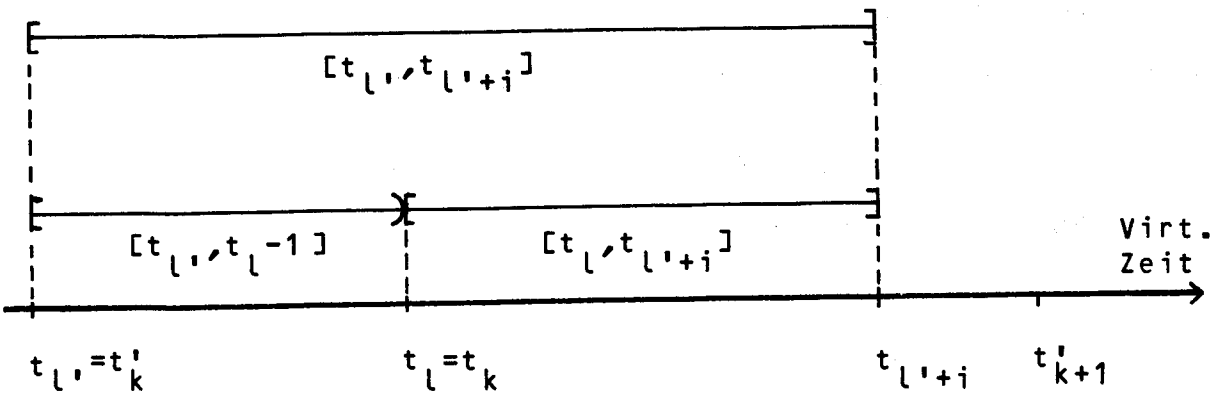
$$t_l = t_k \quad \text{und}$$

$$t_{l'} = t'_k \quad \text{mit } (l, l' \geq k) .$$

Bei beiden Prepaging Verfahren wird das Prepaging spätestens durch die Bedingung (3.2.4) limitiert.

Es sei nun $t'_{k+1} > t_k$ (sonst ist die Satzaussage trivial).

Soll nun die beim OPT-Verfahren zu einem Zeitpunkt $t_{l'+i}$ den i-ten Page Fault nach dem Zeitpunkt $t_{l'}$ verursachende Seite durch Prepaging geladen werden, so muß die gleiche Seite $y_{l'+i}$, die das OPT-Verfahren erst zum Zeitpunkt $t_{l'+i}$ entfernt, bereits zum Zeitpunkt $t_{l'}$ (Verfahren PREOPT') bzw. t_l (Verfahren PREOPT) aus dem Arbeitsspeicher entfernt werden. Die das Prepaging limitierende Bedingung ist, daß die Seite $y_{l'+i}$ im Abschnitt $[t_{l'}, t_{l'+i}]$ bzw. $[t_l, t_{l'+i}]$ der Referenzkette nicht referiert werden darf (vgl. Bild 3.1).



$$L^{t_{l'+i}}_{t_l} = WS(t_{l'+i}, t_{l'+i} - t_l)$$

$$L^{t_{l'+i}}_{t_{l'}} = WS(t_{l'+i}, t_{l'+i} - t_{l'}) = L^{t_l-1}_{t_{l'}} + L^{t_{l'+i}}_{t_l}$$

Bild 3.1
Skizze zur Optimalitaet des PREOPT - Verfahrens

Es muß also

$$y_{l'+i} \notin L_{t_{l'}}^{t_{l'+i}} \quad \text{bzw.}$$

$$y_{l'+i} \notin L_{t_l}^{t_{l'+i}}$$

erfüllt sein.

Da wegen der Voraussetzung $l' \leq l$ ist, gilt

$$L_{t_l}^{t_{l'+i}} \subseteq L_{t_{l'}}^{t_{l'+i}} ;$$

somit wird die Abbruchbedingung mit Sicherheit für das modifizierte Verfahren PREOPT' wirksam, wenn sie beim PREOPT-Verfahren wirksam wird.

$\Longrightarrow$ $t'_{k+1} \geq t_{k+1}$ kann nicht gelten.

$\Longrightarrow$ Es ist

$$t'_{k+1} \leq t_{k+1}$$ q.e.d.

Das PREOPT-Verfahren ist also bzgl. der Fehlseitenrate das beste auf dem OPT-Verfahren aufbauende Prepaging Verfahren.

Unter Zuhilfenahme des Satzes 3.4 läßt sich nun auch die Regularität des PREOPT-Verfahrens zeigen.

Satz 3.5 Das PREOPT-Verfahren verhält sich regulär, d. h. es gilt

FSR (PREOPT, m + 1) ≤ FSR (PREOPT, m).

Beweis: Für die Mengen $\{FS\}$ der Zeitpunkte der Fehlseitenbedingungen gelten aufgrund der Konstruktion des PREOPT-Verfahrens bzw. der Gültigkeit der Einschlußbedingung beim OPT-Verfahren die folgenden Beziehungen:

$$\{FS\,(PREOPT, m+1)\} \subseteq \{FS\,(OPT, m+1)\} \subseteq \{FS\,(OPT, m)\} \quad (3.2.8)$$

und

$$\{FS\ (PREOPT,\ m)\} \subseteq \{FS\ (OPT,\ m)\}\ , \qquad (3.2.9)$$

d. h. die Menge der Fehlseitenbedingungen des OPT(m)-Verfahrens ist eine gemeinsame Übermenge derjenigen der Verfahren PREOPT(m) und PREOPT(m + 1).

Seien $t_k(m)$ und $t_k(m+1)$ die Zeitpunkte der k-ten Fehlseitenbedingung des PREOPT(m) bzw. PREOPT(m + 1)-Verfahrens.

Beh.: Es ist $t_k(m) \leq t_k(m+1)$ für alle k .

Beweis durch vollst. Induktion.

Die Behauptung ist richtig für k = 1, da für beide (alle) Parameterwerte das PREOPT-Verfahren zum Zeitpunkt $t_1(m) = 1$ für alle m die initiale Fehlseitenbedingung liefert.

Beh.: Die Aussage gilt für k + 1, falls sie für k gilt, d. h.

$$t_k(m) \leq t_k(m+1) \Longrightarrow t_{k+1}(m) \leq t_{k+1}(m+1) \qquad (3.2.10)$$

Fallunterscheidung

(a) $t_{k+1}(m) \leq t_k(m+1)$.

Daraus folgt wegen $t_k(m+1) \leq t_{k+1}(m+1)$ sofort

$t_{k+1}(m) \leq t_{k+1}(m+1)$.

(b) $t_k(m) = t_k(m+1)$.

Da die Zeitpunkte $t_k(m)$, $t_k(m+1)$ Page Faults markieren, ist der jeweilige Arbeitsspeicherinhalt (vor dem Prepaging) mit dem des zugrunde liegenden OPT-Verfahrens identisch, d. h.

$S_{t_k-1}\ (PREOPT,\ m) = S_{t_k-1}\ (OPT,\ m)$ $\quad (\equiv S_{t_1-1}\ (OPT,\ m)$, wenn

OPT zum Zeitpunkt t_k die l-te Fehlseitenbedingung liefert)

$$S_{t_k-1}(\text{PREOPT}, m+1) = S_{t_k-1}(\text{OPT}, m+1) \quad (\equiv S_{t_{l'}-1}(\text{OPT}(m+1)).$$

Wegen der Gültigkeit der Einschlußbedingung beim OPT-Verfahren gilt

$$S_{t_k-1}(\text{PREOPT}, m) = S_{t_k-1}(\text{OPT}, m)$$

$$\subseteq S_{t_k-1}(\text{OPT}, m+1)$$

$$= S_{t_k-1}(\text{PREOPT}, m+1) \qquad (3.2.11)$$

Wird nun an der Stelle t_k Prepaging angesetzt, so gilt als Bedingung an die zu ersetzenden Seiten $y_{t_{l+i}}$ $(t_l = t_k, l \geq k)$

$$y_{t_{l+i}}(m) \overset{!}{\in} S_{t_l-1}(m) - L_{t_l}^{t_{l+i}}$$

und

$$y_{t_{l+i}}(m+1) \overset{!}{\in} S_{t_l-1}(m+1) - L_{t_l}^{t_{l+i}}$$

Wegen (3.2.11) gilt

$$S_{t_l-1}(m) - L_{t_l}^{t_{l+i}} \subseteq S_{t_l-1}(m+1) - L_{t_l}^{t_{l+i}} \qquad (3.2.12)$$

Aufgrund der Untermengeneigenschaft (3.2.12) kann die Begrenzung des Prepaging bei Arbeitsspeichergröße $m+1$ nicht eher wirksam werden als bei Größe m. Das gilt erst recht, falls OPT(m) im betrachteten Intervall der Referenzkette eine Fehlseitenbedingung erzeugt die OPT(m + 1) nicht erzeugt, also zusätzlich eine weitere Seite aus dem Arbeitsspeicher entfernt werden muß.

$\Longrightarrow$ $t_{k+1}(m) \leq t_{k+1}(m+1)$, falls $t_k(m) = t_k(m+1)$ ist.

(c) $t_k(m) < t_k(m+1) < t_{k+1}(m)$

Satz 3.4 besagt: Gilt für zwei Startwerte $t_k, t_k^+ \in \{FS(OPT, m)\}$ eines Prepaging Schrittes nach dem PREOPT-Verfahren

$$t_k \leq t_k^+,$$

dann gilt auch

$$t_{k+1} \leq t_{k+1}^+ .$$

Wegen (3.2.8) und (3.2.9) gilt

$$t_k(m), t_k(m+1) \in \{FS(OPT(m)\}.$$

Damit ist Satz 3.4 mit $t_k^+(m) = t_k(m+1)$ anwendbar und liefert

$$t_{k+1}(m) \leq t_{k+1}^+(m) .$$

Ferner liefert dann die Aussage nach Fall (b)

$$t_{k+1}^+(m) \leq t_{k+1}(m+1).$$

$$\Longrightarrow t_{k+1}(m) \leq t_{k+1}(m+1).$$

$\Longrightarrow$ Die Aussage (3.2.10) ist richtig.

$\Longrightarrow$ $t_k(m) \leq t_k(m+1)$ für alle k.

$\Longrightarrow$ FSR (PREOPT, m + 1) $\leq$ FSR (PREOPT, m) q.e.d.

Wie oben gezeigt, verhält sich das PREOPT-Verfahren regulär, es genügt jedoch - wie sich leicht zeigen läßt - keiner Einschlußbedingung:

PREOPT(m) liefert die zweite Fehlseitenbedingung (nach der initialen zum Zeitpunkt $t_1(m) = 1$) genau an der Stelle, an der die (m + 1)-te verschiedene Seite referiert wird zum Zeitpunkt $t_2(m)$. PREOPT(m + 1) liefert an dieser Stelle keine Fehlseitenbedingung, sondern an der Stelle $t_2(m+1)$, an der die (m + 2)-te Seite referiert wird. An dieser Stelle würde PREOPT(m) aber nur dann ebenfalls eine Fehlseitenbedingung

erzeugen, wenn zum Zeitpunkt $t_2(m)$ kein Prepaging möglich war, wovon i. a. nicht ausgegangen werden kann. Die Menge $\{t_k(m)\} = \{FS\ (PREOPT, m)\}$ ist demnach keine Übermenge von $\{t_k(m+1)\} = \{FS\ (PREOPT, m+1)\}$ woraus folgt

$$S_t\ (PREOPT, m) \notin S_t\ (PREOPT, m+1)\ .$$

3.2.1.3 Das OPR - Verfahren

Das OPR-Verfahren organisiert bei jeder Fehlseitenbedingung den Arbeitsspeicherinhalt so, daß die Zeit bis zur nächstfolgenden Fehlseitenbedingung maximiert wird. Dies wird durch

$$S_t(m) = L_t^{j(t,\,m)}$$

erreicht mit

$$j(t, m) = \max_j\ (|L_t^j| = m)\ .$$

Das Verfahren soll nun durch Festlegung der Mengen X_t, Y_t beschrieben werden, woraus sich durch die Beziehung

$$S_t = S_{t-1} + X_t - Y_t$$

die Vorschrift ergibt, wie aus einem Arbeitsspeicherinhalt der nächste zu bilden ist.

$$X_t = L_t^{j(t,m)} - S_{t-1} = K_t^{j(t,m)}$$

$$Y_t = S_{t-1} - L_t^{j(t,m)} = S_{t-1} - (L_t^{j(t,m)} - K_t^{j(t,m)}) \qquad (3.2.13)$$

für alle t, für die $r_t \notin S_{t-1}$ ist,

d. h., es werden von den nächsten m verschiedenen Seiten nach dem Zeitpunkt t (einschließlich) diejenigen geladen, die nicht bereits im Arbeitsspeicher sind, und entladen werden solche Seiten aus S_{t-1}, die nicht in $L_t^{j(t,m)}$ enthalten sind, wovon für die Differenzmengenbildung jedoch nur die Elemente aus $L_t^{j(t,m)}$ relevant sind, die in S_{t-1} enthalten sind, d. h. $L_t^{j(t,m)} - K_t^{j(t,m)}$. Wie leicht zu sehen ist, ist mit (3.2.13)

$$S_t = S_{t-1} + X_t - Y_t = L_t^{j(t, m)}$$

wie ursprünglich gefordert.

Satz 3.6 Das Seitenaustauschverfahren OPR ist in der Klasse der Verfahren fester Arbeitsspeichergröße bezüglich der Fehlseitenrate optimal, d. h. es ist

$$FSR(OPR, m, \omega) \leq FSR(A, m, \omega)$$

für beliebige Seitenaustauschalgorithmen A, beliebige Referenzketten ω und jedes m.

Beweis: Es wird gezeigt, daß OPR in jedem Abschnitt $r_1 \ldots r_t$ einer Referenzkette weniger Fehlseitenbedingungen liefert als jedes andere Verfahren, d.h. es gilt

$$FS_1^t(OPR, m, \omega) \leq FS_1^t(A, m, \omega),$$

wobei FS_1^k die Anzahl der Fehlseitenbedingungen im Abschnitt $r_1 \ldots r_k$ $(1 \leq k)$ der Referenzkette bezeichnet; $FS_1^k = 0$ für $1 > k$.

Der Beweis erfolgt durch vollst. Induktion.

Seien $1, t_1, t_2, \ldots, t_k$ mit $t_k \leq t < t_{k+1}$
zeitlich geordnet die Zeitpunkte, an denen das OPR-Verfahren Fehlseitenbedingungen erzeugt.

Dann ist

$$FS_1^{t_k}(OPR) = FS_1^t(OPR)$$

$$\Longrightarrow \quad 0 = FS_{t_k+1}^t(OPR) \leq FS_{t_k+1}^t(A), \qquad (3.2.14)$$

da kein Verfahren in einem Abschnitt der Referenzkette weniger als 0 Fehlseitenbedingungen erzeugen kann.

Ferner gilt für beliebige Verfahren

$$FS_1^t(A) = FS_1^{t_k}(A) + FS_{t_k+1}^{t}(A). \qquad (3.2.15)$$

Induktionsanfang:

Es ist zu zeigen, daß

$$FS_1^{t_1}(OPR, m) \leq FS_1^{t_1}(A, m)$$

gilt.

Nach der Definition des Verfahrens OPR ist

$$S_1(OPR, m) = L_1^{j(1,m)} = L_1^{t_1-1},$$

d. h. der Abschnitt $r_1 \ldots r_{t_1-1}$ enthält die ersten m verschiedenen Seiten der Referenzkette und genau an der Stelle t_1 wird die (m + 1)-te Seite referiert.

Wenn es ein Verfahren A gäbe, das die zweite Fehlseitenbedingung (wegen $S_o = \emptyset$ liefert r_1 für alle Verfahren die erste Fehlseitenbedingung!) an einer späteren Stelle als t_1 erzeugen würde, so müßte dieses Verfahren in einem Arbeitsspeicher der Größe m Seitenrahmen mindestens m + 1 verschiedene Seiten unterbringen können. Dies ist unmöglich. Daraus folgt, daß jeder Austauschalgorithmus bis zum Zeitpunkt t_1 wenigstens die zweite Fehlseitenbedingung liefern muß.

$$2 = FS_1^{t_1}(OPT, m) \leq FS_1^{t_1}(A, m).$$

Induktionsvoraussetzung:

$$FS_1^{t_1}(OPR, m) \leq FS_1^{t_1}(A, m). \qquad (3.2.16)$$

Beh.: Aus (3.2.16) folgt

$$FS_1^{t_{l+1}}(OPR, m) \leq FS_1^{t_{l+1}}(A, m).$$

Beh.: $FS_{t_l+1}^{t_{l+1}}(OPR, m) \leq FS_{t_l+1}^{t_{l+1}}(A, m).$

Der Beweis erfolgt analog zum Beweis der Induktionsvoraussetzung. Gemäß der Definition des Verfahrens OPR erzeugt $r_{t_{l+1}}$ die erste Fehlseitenbedingung nach der Referenz r_{t_l}, da $r_{t_{l+1}}$ genau die $(m+1)$-te verschiedene Seite im Abschnitt $r_{t_l} \ldots r_{t_{l+1}}$ ist, muß auch jedes andere Verfahren mindestens eine Fehlseitenbedingung in diesem Abschnitt der Referenzkette erzeugen.

$$\Longrightarrow \quad 1 = FS_{t_l+1}^{t_{l+1}}(OPR, m) \leq FS_{t_l+1}^{t_{l+1}}(A, m).$$

Addiert man darauf die Induktionsvoraussetzung (3.2.16) so erhält man wegen (3.2.15)

$$FS_1^{t_{l+1}}(OPR) = FS_1^{t_l}(OPR) + FS_{t_l+1}^{t_{l+1}}(OPR)$$

$$\leq FS_1^{t_l}(A) + FS_{t_l+1}^{t_{l+1}}(A)$$

$$= FS_1^{t_{l+1}}(A).$$

$\Longrightarrow$ Für bel. Page Fault Zeitpunkte t_l des OPR-Verfahrens gilt

$$FS_1^{t_l}(OPR) \leq FS_1^{t_l}(A).$$

Insbesondere gilt damit

$$FS_1^{t_k}(OPR) \leq FS_1^{t_k}(A).$$

Hieraus folgt durch Addition von (3.2.14) wegen (3.2.15)

$$FS_1^t(OPR) = FS_1^{t_k}(OPR) + FS_{t_k+1}^{t}(OPR)$$

$$\leq FS_1^{t_k}(A) + FS_{t_k+1}^{t}(A)$$

$$= FS_1^t(A).$$

q.e.d.

Satz 3.7 Das OPR-Verfahren verhält sich regulär, d. h., es gilt

$FSR(OPR, m+1) \leq FSR(OPR, m)$.

Beweis: Das OPR-Verfahren teilt durch die von ihm erzeugten Fehlseitenbedingungen die Referenzkette in maximale Abschnitte, in denen jeweils m verschiedene Seiten referiert werden.

Seien $t_k(m)$ und $t_k(m+1)$ die Zeitpunkte der k-ten Fehseitenbedingung von OPR(m) bzw. OPR(m + 1).

Beh.: $t_k(m) \leq t_k(m+1) \Rightarrow t_{k+1}(m) \leq t_{k+1}(m+1)$. (3.2.17)

(a) Sei zunächst

$$t_k(m) = t_k(m+1).$$

Laut Definition ist

$$S_{t_k}(m) = L_{t_k}^{j(t_k,m)} \quad \text{mit } j(t_k, m) + 1 = t_{k+1}(m)$$

und

$$S_{t_k}(m+1) = L_{t_k}^{j(t_k,m+1)} \quad \text{mit } j(t_k, m+1) + 1 = t_{k+1}(m+1).$$

Da die durch $L_{t_k}^{j(t_k,m)}$ und $L_{t_k}^{j(t_k,m+1)}$ beschriebenen Intervalle

der Referenzkette bei gleichem Anfangspunkt m bzw. m + 1 verschiedene Referenzen enthalten, muß für die Intervallendpunkte

$$j(t_k, m + 1) > j(t_k, m)$$

gelten.

$$\Longrightarrow \quad t_{k+1}(m) = j(t_k,m) + 1 < j(t_k, m + 1) + 1 = t_{k+1}(m + 1).$$

(b) Sei nun $t_k^+(m) < t_k(m + 1)$.

Dazu werde zunächst ein Verfahrensschritt von OPR(m) für

$$t_k^+(m) < t_k(m)$$

betrachtet (vgl. auch Bild 3.2).

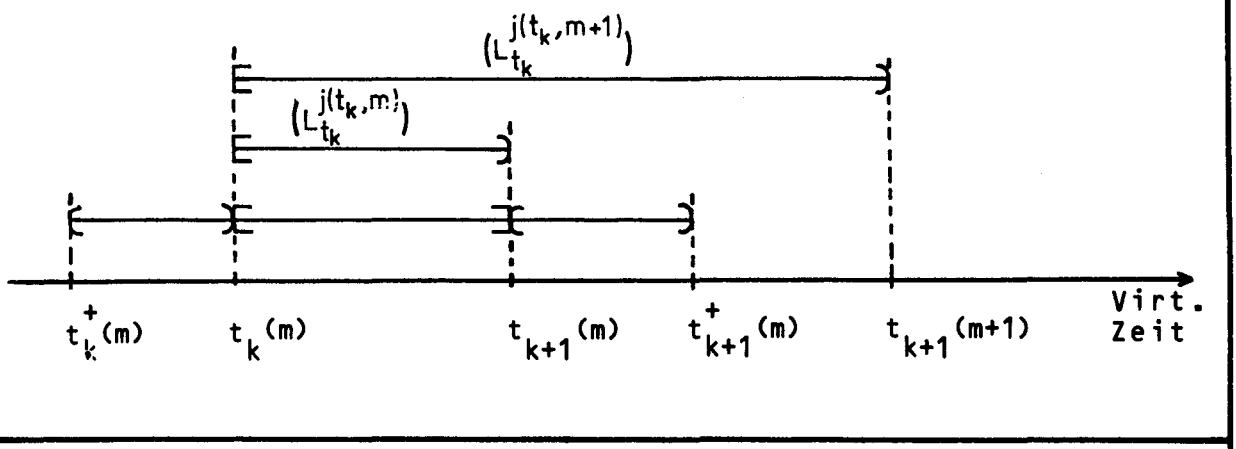


Bild 3.2
Skizze zur Regularitaet des OPR - Verfahrens

Annahme: $t_{k+1}^+(m) > t_{k+1}(m)$.

Aufgrund der Annahme kann das durch $L_{t_k^+}^{j(t_k^+, m)}$ beschriebene

Intervall der Referenzkette wie folgt zusammengesetzt werden:

$$[t_k^+, t_{k+1}^+) = [t_k^+, t_k) + [t_k, t_{k+1}] + (t_{k+1}, t_{k+1}^+)],$$

wobei bereits $[t_k, t_{k+1}]$ wegen der Definition von $L_{t_k}^{j(t_k,m)}$ genau m + 1 verschiedene Elemente enthält.

$\Longrightarrow$ Widerspruch zu $\left|L_{t_k^+}^{j(t_k^+, m)}\right| = m$;

$\Longrightarrow$ die Annahme ist falsch, es gilt

$$t_{k+1}^+(m) \leq t_{k+1}(m),$$

$\Longrightarrow$ falls $t_k(m) = t_k(m + 1)$ ist wegen (a)

$t_{k+1}(m) < t_{k+1}(m + 1)$, so daß insgesamt gilt:

$t_k^+(m) < t_k(m + 1) \Longrightarrow t_{k+1}^+(m) < t_{k+1}(m + 1)$.

$\Longrightarrow$ Es gilt immer (3.2.17), womit die Aussage des Satzes 3.7 bewiesen ist.

Wie man sich leicht klar macht, sind die Intervalle, in die die Referenzkette durch die Fehlseitenbedingungen von OPR(m) und OPR(m + 1) eingeteilt wird (vgl. Bild 3.3) für $1 < m < n - 1$ im allgemeinen nicht ineinandergeschachtelt, d. h. das OPR-Verfahren erfüllt keine Einschlußbedingung.

Das OPR-Verfahren ist identisch mit dem von TRIVEDI /T2/ unabhängig von dieser Arbeit vorgestellten DPMIN-Verfahren.

Da die beiden Prepaging Verfahren PREOPT und OPR einer Einschlußbedingung nicht genügen, können vereinfachte Verfahren zur Simulation analog zu dem in Kap. 2.6.1 für das LRU-Verfahren hergeleiteten nicht zur Anwendung kommen. Die Simulation ist somit aufwendig, da sie für jede Vorgabe von m durchgeführt werden muß.

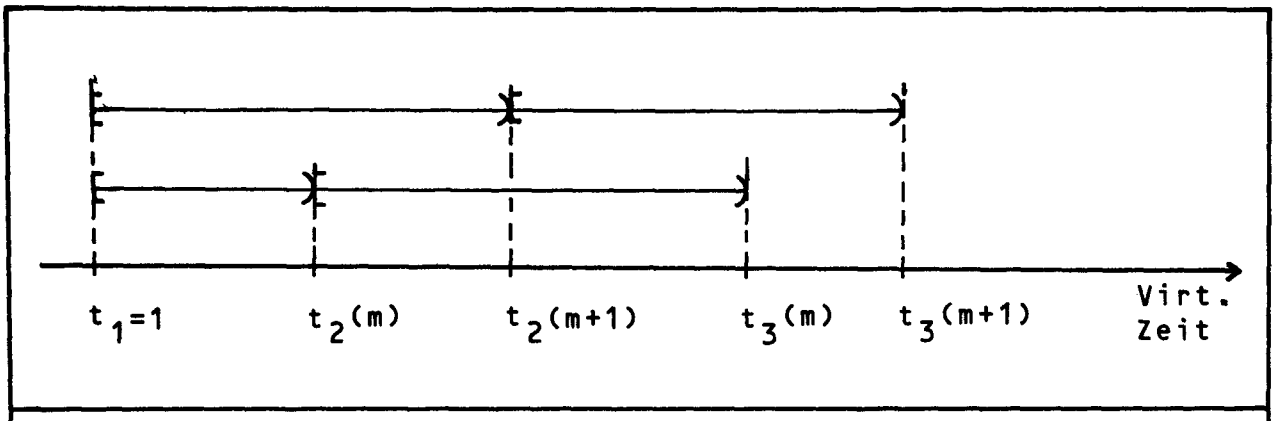


Bild 3.3
Ungueltigkeit der Einschlussbedingung beim
OPR - Verfahren

3.2.2 Verfahren variabler Areitsspeichergröße

In diesem Kapitel wird zunächst ein zu Satz 3.1 analoger Satz bewiesen, der die Aussage dieses Satzes, daß durch Prepaging die Anzahl der Ladevorgänge nicht reduziert werden kann, auf Verfahren variabler Arbeitsspeichergröße erweitert. Danach werden zwei auf dem VMIN-Verfahren basierende Prepaging Verfahren vorgestellt, von denen das eine, VMINPP1, gedanklich dem VMIN-Konzept nahesteht, während das zweite, VMINPP2, sich an das PFF-Konzept anlehnt.

3.2.2.1 Allgemeines

Satz 3.8 Zu jedem Paging Verfahren A existiert ein Demand Paging Verfahren A', das bei nicht größerer mittlerer Working Set Größe nicht mehr Seiten lädt als A, d. h. für das

$$X(A', \overline{m}', \omega) \leq X(A, \overline{m}, \omega) \text{ mit } \overline{m}' \leq \overline{m}$$

gilt für alle ω.

Beweis: Analog zu Satz 3.1 erfolgt der Beweis durch Konstruktion eines Algorithmus A' mit der geforderten Eigenschaft.

Der Algorithmus A' wird so gebildet, daß er Seiten nur dann lädt, wenn sie eine Fehlseitenbedingung erzeugen, Seiten jedoch immer aus dem Arbeitsspeicher entfernt, wenn sie durch A entfernt werden (sofern sie überhaupt in S(A') enthalten sind).

Sei P_t die Menge der zum Zeitpunkt t von A aber nicht von A' geladenen Seiten (zurückgestellte Ladevorgänge), dann gilt

$$P_t = P_{t-1} - Y_t + X_t - (\{r_t\} - S'_{t-1}),$$

d. h., eine Seite ist genau dann zum Zeitpunkt t ein zurückgestellter Ladevorgang, wenn sie es entweder schon zum Zeitpunkt t - 1 war und durch den Algorithmus A nicht zum Zeitpunkt t aus dem Arbeitsspeicher entfernt wurde oder zum Zeitpunkt t durch A, nicht aber durch A' geladen wurde. Bei Demand Prepaging (d. h. Prepaging wird nur zu Page Fault Zeitpunkten initialisiert) - wie es in dieser Arbeit grundsätzlich unterstellt wird - ist

$$P_t = P_{t-1} + X_t - \{r_t\} - Y_t .$$

Es gilt für das Verfahren A

$$S_t = S_{t-1} + X_t - Y_t$$

und für A'

$$S'_t = S'_{t-1} + X'_t - Y'_t$$

mit

$$X'_t = \begin{cases} \emptyset, \text{ falls } r_t \in S'_{t-1} \\ \{r_t\} \text{ sonst} \end{cases}$$

und

$$Y'_t = Y_t \cap S'_{t-1},$$

woraus insbesondere folgt, daß $Y'_t = \emptyset$ ist, falls $Y_t = \emptyset$ ist.

Es werden somit alle Seiten aus dem Arbeitsspeicher (A') entfernt, die aus dem Arbeitsspeicher (A) entfernt werden, sofern sie enthalten sind; geladen werden von den Seiten, die A lädt, nur die, die wirklich referiert werden und zwar zum Zeitpunkt der Referenz.

$$\Longrightarrow S'_t = S_t - P_t;$$

$$\Longrightarrow S'_t \subseteq S_t \quad \text{für alle } t;$$

$$\Longrightarrow \overline{m}(A') = \overline{m}' \leq \overline{m} = \overline{m}(A). \qquad (3.2.18)$$

Ferner gilt für jede Seite, die A' lädt, daß sie entweder auch zum gleichen Zeitpunkt von A geladen wird ($r_t \notin S_{t-1}$) oder daß sie bereits zu einem früheren Zeitpunkt geladen worden ist ($r_t \in P_{t-1}$).

$$\Longrightarrow \sum_{i=1}^{t} |X'_j| \leq \sum_{i=1}^{t} |X_i|$$

$$\Longrightarrow X(A', \overline{m}', \omega) \leq X(A, \overline{m}, \omega),$$

wobei wegen (3.2.18) $\overline{m}' \leq \overline{m}$ ist. q.e.d.

3.2.2.2 Das VMINPP1 - Verfahren

Das VMIN-Verfahren beruht auf einer Kostenäquivalenz zwischen dem Laden einer Seite und dem (unreferierten) Verbleib im Arbeitsspeicher für eine bestimmte virtuelle Zeit T.

Mit der gleichen Überlegung kann auch eine Wertäquivalenz zwischen einer Fehlseitenbedingung und einer Aufenthaltsdauer T' im Arbeitsspeicher etabliert werden.

Auf dieser Basis ist das VMINPP1-Verfahren definiert. Durch vorzeitiges Laden der als nächste eine Fehlseitenbedingung verursachenden Seite wird ein Page Fault gegenüber dem VMIN-Verfahren eingespart; dafür befindet sich die vorzeitig geladene Seite zusätzlich T* Zeiteinheiten unreferiert im Arbeitsspeicher.

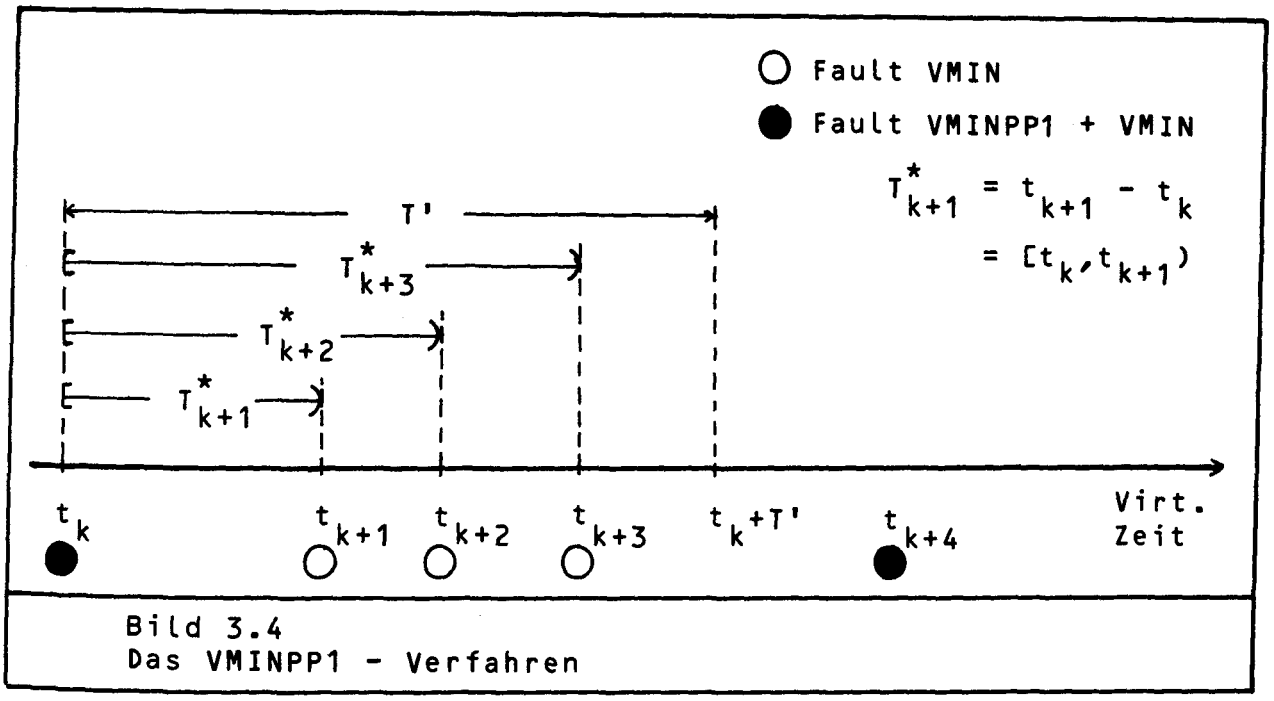


Bild 3.4
Das VMINPP1 - Verfahren

Das VMINPP1-Verfahren ist wie folgt definiert (vgl. auch Bild 3.4)

$$S_t(\text{VMINPP1}) = S_{t-1}(\text{VMINPP1}) + X_t - Y_t$$

mit

$$X_t = \begin{cases} \emptyset, \text{ falls } r_t \in S_{t-1} \\ L_t^{t+T'} \cap S_{t-1} = K_t^{t+T'} \end{cases}$$

$$Y_t = Y_t(\text{VMIN}),$$

d. h. eine Seite wird genau dann durch Prepaging geladen, wenn sie andernfalls innerhalb der nächsten T' Zeiteinheiten eine Fehlseitenbedingung erzeugen würde; beim VMINPP1-Verfahren haben aufeinanderfolgende Fehlseitenbedingungen somit einen

Mindestabstand von T' Zeiteinheiten.

Objektiv aber auch aufgrund der obigen Definition ist das Verfahren nur sinnvoll für $T' \leq T$, wobei T der Zeitparameter des zugrunde liegenden VMIN-Verfahrens ist. $T' > T$ kann aber zugelassen werden, wenn eine ordnungsgemäße Behandlung der dann möglichen Sonderfälle sichergestellt ist:

(a) Durch Prepaging geladene Seiten können zwischen Ladezeitpunkt und Zeitpunkt der Programmreferenz länger als T Zeiteinheiten unreferiert im Arbeitsspeicher sein.

(b) In dem durch Prepaging abgedeckten Intervall kann beim zugrundeliegenden VMIN-Verfahren dieselbe Seite mehr als einen Page Fault verursachen.

Im gleichen Sinne, wie das VMIN-Verfahren kostenoptimal ist, da es an den Entscheidungspunkten für jede Seite die jeweils kostengünstigere Alternative wählt, ist auch das VMINPP1-Verfahren optimal.

Satz 3.9 Das VMINPP1-Verfahren verhält sich bei vorgegebenem VMIN(T) bezüglich des Verfahrensparameters T' nicht regulär.

Beweis: (a) Das Verfahren erfüllt die Regularitätsbedingung (2.5.3), d.h. es gilt:

$$T' \leq \overline{T}' \Rightarrow FSR(\overline{T}') \leq FSR(T')$$

Das VMINPP1-Verfahren teilt durch seine Page Faults die Referenzkette in Abschnitte der Länge $\geq T'$, wobei der erste Page Fault nach $t_k + T'$ (vgl. Bild 3.4) das zum Zeitpunkt t_k beginnende Intervall abschließt. Ist nun $T' \leq \overline{T}'$, so ist klar, daß Intervallendpunkte beim Parameterwert $\overline{T}'$ niemals vor Intervallendpunkten mit gleichem Index des Parameterwertes T' liegen können.

$\Longrightarrow$ VMINPP1($\overline{T}'$) erzeugt höchstens so viele Page Faults wie VMINPP1(T').

(b) Das Verfahren VMINPP1 erfüllt nicht die Parameter-Regularitätsbedingung (2.5.4) und nicht die Speicher-Regularitätsbedingung (2.5.2).

Dazu werde das in Bild 3.5 gegebene Beispiel betrachtet. Die Größe von T im Verhältnis zu T' = 7 und $\overline{T}'$ = 11 spielt keine Rolle, da in dem gegebenen Beispiel keine Seite nach der letzten Referenz in direkter Folge erneut referert wird.

Aus diesem Beispiel ergibt sich

$\overline{m}(T' = 7) = \frac{35}{20}$ und $\overline{m}(T' = 11) = \frac{33}{20}$,

d. h. Anomalie im Sinne der Bedingung (2.5.4),

und

$FSR(\overline{m} = \frac{35}{20}) = \frac{3}{20}$ und

$FSR(\overline{m} = \frac{33}{20}) = \frac{2}{20}$,

d. h. Anomalie im Sinne der Bedingung (2.5.2.).

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ω	1	1	1	1	1	1	1	2	2	2	2	3	4	5	6	6	6	6	6	6
S_t(VMIN)	①	1	1	1	1	1	1	②	2	2	2	③	④	⑤	⑥	6	6	6	6	6
S_t(VMINPP1) T'=7	①	1	1	1	1	1	1	② 3 4 5	2 3 4 5	2 3 4 5	2 3 4 5	3 4 5	4 5	5	⑥	6	6	6	6	6
S_t(VMINPP1) T'=11	① 2	1 2	1 2	1 2	1 2	1 2	1 2	2	2	2	2	③ 4 5 6	4 5 6	5 6	6	6	6	6	6	6

Bild 3.5
Beispiel fuer anomales Verhalten des VMINPP1 - Verfahrens

Da bei Prepaging Ladezeitpunkt und Zeitpunkt der ersten Referenz auf eine Seite nach dem Laden auseinanderfallen, muß die Bedeutung des Zeitparameters T neu überdacht werden (vgl. Bild 3.6).

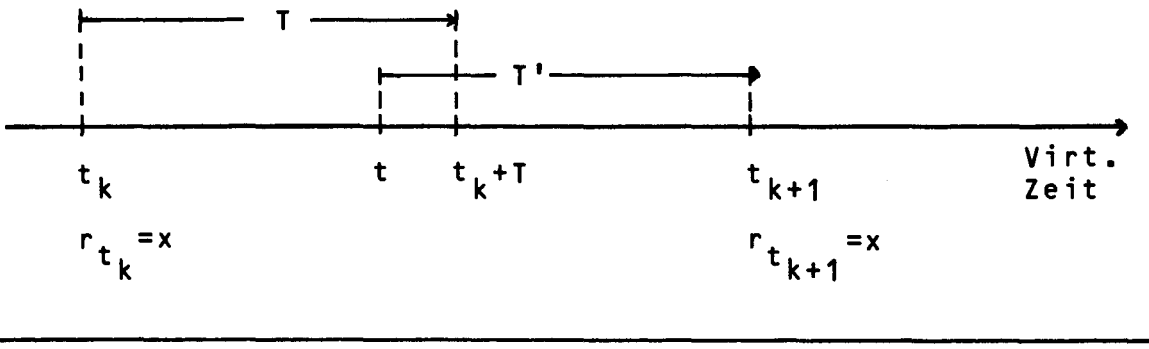


Bild 3.6
Bedeutung des Zeitparameters T bei Prepaging-Verfahren

Seien t_k und t_{k+1} aufeinanderfolgende Referenzen auf die Seite x und der Abstand größer T (d. h. $d_{t_k}(x) = t_{k+1} - t_k > T$). Beim VMIN-Verfahren wird die Seite x wegen $t_{k+1} - t_k > T$ unmittelbar nach der Referenz zum Zeitpunkt t_k aus dem Arbeitsspeicher entfernt und durch Page Fault zum Zeitpunkt t_{k+1} erneut geladen. Sei nun t Zeitpunkt einer Fehlseitenbedingung des VMINPP1-Verfahrens, dann wird die Seite x, da sie wegen $t_{k+1} - t \leq T'$ im nächsten T'-Intervall eine Fehlseitenbedingung erzeugen würde, bereits zum Zeitpunkt t durch Prepaging geladen. Damit aber geschieht der nächste Zugriff (im Sinne des Ladens) auf die Seite x nach dem Zeitpunkt t_k bereits zum Zeitpunkt t und damit wegen $t - t_k < T$ nach weniger als T Zeiteinheiten.

Es ist also festzulegen, ob sich das Zeitfenster T nur auf Programmreferenzen oder auch auf Ladezugriffe beziehen soll. Für beide Alternativen gibt es Argumente. Wenn auch Ladezugriffe gewertet werden, so entsteht ein Widerspruch zum ursprünglichen Verfahren, da eine Seite länger als T Zeiteinheiten unreferiert im Arbeitsspeicher

verweilen kann (bei Konstellation wie in Bild 3.6 dargestellt wird der Parameterwert T durch T + T' ersetzt). Für diese Auslegung spricht, daß eine Seite - was sonst beim VMIN-Verfahren nicht auftreten kann - für beliebig kurze Zeit aus dem Arbeitsspeicher entfernt werden kann.
Für die Beibehaltung der ursprünglichen Bedeutung spricht, daß sich dann kein Unterschied in der Logik der Fenstergröße gegenüber dem VMIN-Verfahren ergibt und die Gesamtzahl der Sekundärspeicherzugriffe sich ja nicht ändert.

Bei den Simulationen wurden beide Alternativen durchgeführt, wobei sich erwartungsgemäß bei Berücksichtigung der Ladezeitpunkte eine Reduktion der Ladevorgänge und Fehlseitenbedingungen bei gleichzeitiger Vergrößerung der mittleren Speicherbelegung ergab. Für die Simulationen wurde T' = T gewählt.

Es liegt auf der Hand, daß das VMIN-Verfahren, bei dem für jede einzelne Seite eine optimale lokale Entscheidung getroffen wird, global nicht optimal ist.
Bei Verfahren globaler Optimierung könnten zwei Strategien verfolgt werden: zum einen könnte die Kostenfunktion des VMIN-Verfahrens global minimiert werden, zum andern könnte die Anzahl der Page Faults minimiert werden, ohne daß die durch das VMIN-Verfahren vorgegebenen Kosten überschritten werden. In beiden Fällen ergeben sich so aufwendige Optimierungsaufgaben, daß eine praktische Verwertbarkeit, und sei es auch nur als Referenzverfahren, ausgeschlossen ist.

3.2.2.3 Das VMINPP2 - Verfahren

Auf der Basis des VMIN-Verfahrens läßt sich ein zweites Prepaging Verfahren definieren, VMINPP2 genannt, das in seiner Motivation dem PFF - Verfahren nahesteht.

Ist es zum einen erklärtes Ziel aller Seitenaustauschverfahren, genau die zu einer Programmphase gehörende Lokalität im Arbeitsspeicher zu halten, so ist es sicher auch

sinnvoll zu versuchen, die zu einer Lokalität gehörenden Seiten auf einmal zu laden und erst dann die Ausführung des Programms fortzusetzen.

Unterstellt man nun wiederum ein Phasen-/Übergangsmodell des Programmverhaltens, so sind Häufungen von Fehlseitenbedingungen (d. h. kurze Abstände aufeinander folgender Fehlseitenbedingungen) ein Zeichen für den Aufbau einer neuen Lokalität. Aufgrund dieser Überlegungen wird das VMINPP2-Verfahren wie folgt definiert (vgl. Bild 3.7):

$$S_t(\text{VMINPP2}, T') = S_{t-1}(\text{VMINPP2}, T') + X_t - Y_t$$

mit

$$X_t = \begin{cases} \emptyset, \text{ falls } r_t \in S_{t-1} \\ L_t^{t+i_o} \cap S_{t-1} = K_t^{t+i_o} \text{ sonst} \end{cases}$$

$$Y_t = Y_t(\text{VMIN})$$

mit $i_o = \max(i | t_{k+1} - t_{k+i-1} \leq T')$, wenn $t = t_k$ ist, wobei t_k den Zeitpunkt der k-ten Fehlseitenbedingung des zugrunde liegenden VMIN-Verfahrens bezeichnet.

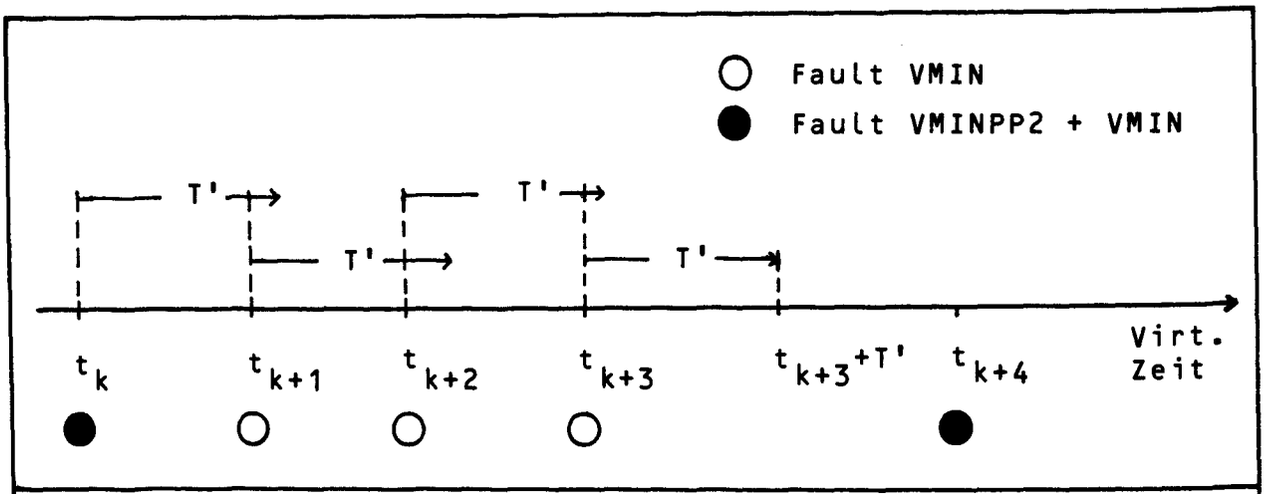


Bild 3.7
Das VMINPP2 - Verfahren

Das VMINPP2-Verfahren verhindert durch Prepaging also solche Fehlseitenbedingungen des VMIN-Verfahrens, deren zeitlicher Abstand voneinander einen vorgegebenen Grenzwert T' nicht überschreitet. Somit garantiert auch das VMINPP2-Verfahren einen zeitlichen Mindestabstand aufeinanderfolgender Fehlseitenbedingungen. Anders als beim VMINPP1-Verfahren ist dem durch Prepaging überdeckten Zeitabschnitt der Referenzkette keine obere Schranke gesetzt; durch eine Bedingung an T' kann nicht sichergestellt werden, daß das durch Prepaging überdeckte Intervall die Größe des VMIN-Parameters T nicht übersteigt. Dies kann jedoch nur bei unsachgemäßer Wahl des Parameters T' oder auch des Parameters T des VMIN-Verfahrens geschehen.

Wie auch beim VMINPP1-Verfahren entstehen 2 Varianten dieses Verfahrens dadurch, daß beim Zeitparameter T des VMIN-Verfahrens einmal Ladezugriffe (beim Prepaging) berücksichtigt werden und einmal nicht.

Satz 3.10 Bei festem Parameter T des zugrundeliegenden VMIN-Verfahrens genügt das VMINPP2-Verfahren bezüglich des Parameters T' einer Einschlußbedingung und verhält sich deshalb regulär.

Beweis: Da das VMINPP2-Verfahren Page Faults des VMIN-Verfahrens vermeidet, indem die Fault erzeugenden Seiten bereits vorher durch Prepaging geladen werden, gilt

$$\{FS(VMINPP2)\} \subseteq \{FS(VMIN)\} \text{ für beliebige Parameter T' des VMINPP2-Verfahrens.}$$

Ist nun t_k Zeitpunkt eines gemeinsamen Page Faults, so gilt

$$S_{t_k^-}(VMINPP2) = S_{t_k^-}(VMIN)$$

(t_k^- heißt: bevor die Aktionen zum Zeitpunkt t_k durchgeführt sind).

Sei nun t_k Zeitpunkt eines gemeinsamen Page Faults von VMINPP2($\bar{T}'$) und

VMINPP2(T') und damit auch von VMIN, dann gilt

$$S_{t_k}\text{-}(VMIN) = S_{t_k}\text{-}(VMINPP2, \overline{T}') = S_{t_k}\text{-}(VMINPP2, T').$$

Die das Prepaging begrenzende Bedingung ist

$$t_{k+i} - t_{k+i-1} > T' \qquad (3.2.20)$$

bzw.

$$t_{k+j} - t_{k+j-1} > \overline{T}', \qquad (3.2.21)$$

wobei wegen $T' \leq \overline{T}'$ (3.2.20) immer erfüllt ist, wenn (3.2.21) gilt.

$\Longrightarrow$ VMINPP2($\overline{T}'$) lädt zum Zeitpunkt t_k nicht weniger Seiten durch Prepaging als VMINPP2(T').

$$\Longrightarrow S_{t_k}(VMINPP2, T') \subseteq S_{t_k}(VMINPP2, \overline{T}') \qquad (3.2.22).$$

Da das VMINPP2-Verfahren Seiten unabhängig vom Parameterwert genau zu den Zeitpunkten des zugrundeliegenden VMIN-Verfahrens entfernt, bleibt (3.2.22) gültig bis eine Page Fault Bedingung eintritt, was wegen (3.2.22) entweder wiederum eine gemeinsame Fehlseitenbedingung ist oder nur eine Fehlseitenbedingung von VMINPP2(T') ist.

Ist nun t_{k+i} Page Fault Zeitpunkt des Verfahrens VMINPP2(T') alleine, so wird ein Prepaging-Schritt für VMINPP2(T') durchgeführt. Da jedoch aus der Gültigkeit von (3.2.21) die Gültigkeit von (3.2.20) folgt, kann dieser Schritt nicht über einen Page Fault Zeitpunkt von VMINPP2 ($\overline{T}'$) hinausgehen (vgl. Bild 3.8).

$\Longrightarrow$ VMINPP2(T') lädt zum Zeitpunkt t_{k+i} nur solche Seiten in den Arbeitsspeicher, die VMINPP2($\overline{T}'$) bereits zum Zeitpunt t_k geladen hat.

$\Longrightarrow$ $S_t(\text{VMINPP2}, T') \subseteq S_t(\text{VMINPP2}, \bar{T}')$ für $t = t_k \ldots t_{k+j}^-$,

falls t_{k+j} der nächste Zeitpunkt einer gemeinsamen Fehlseitenbedingung nach t_k ist.

$\Longrightarrow$ $S_t(\text{VMINPP2}, T') \subseteq S_t(\text{VMINPP2}, \bar{T}')$, falls $T' \leq \bar{T}'$ ist.

q.e.d.

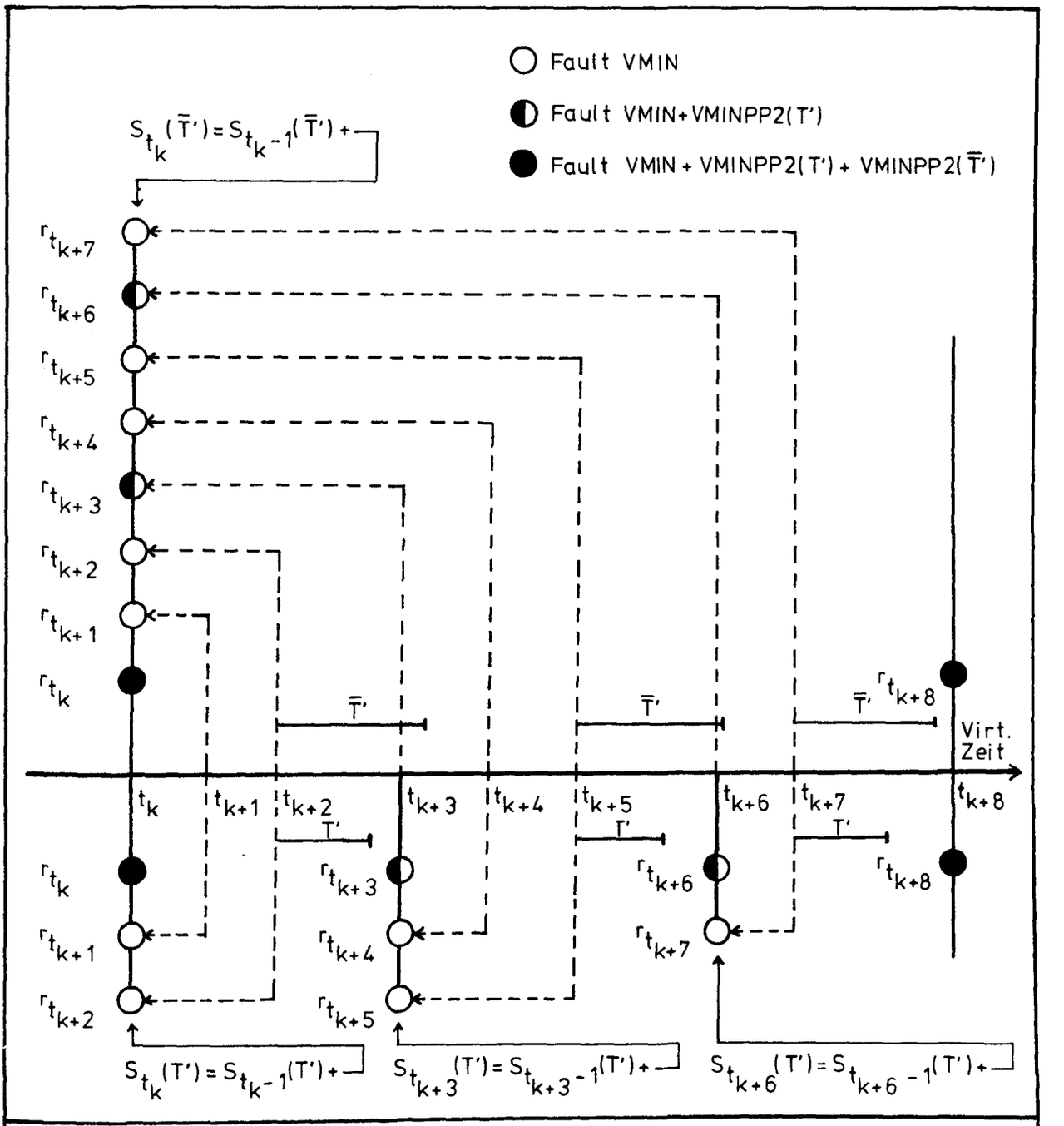


Bild 3.8
Einschlusseigenschaft beim VMINPP2 - Verfahren

3.3. Nichtoptimale Prepaging Verfahren

3.3.1 Motivation - Allgemeines

Seit es Systeme mit seitenorientierten virtuellen Speichern gibt, existiert das Problem, die Anzahl der Fehlseitenbedingungen gegen die Größe des im Mittel belegten Arbeitsspeichers auszubalancieren. Häufige Fehlseitenbedingungen erhöhen den Systemoverhead beträchtlich und können im Extremfall sogar durch Überbeanspruchung des Sekundärspeichers zu 'Thrashing' führen; bei einer sehr geringen Anzahl von Fehlseitenbedingungen ist der vorhandene reale Arbeitsspeicher unökonomisch eingesetzt.

Bei Betrachtung der Fehlseitenrate in Abhängigkeit der Arbeitsspeichergröße ist festzustellen, daß unterhalb einer kritischen Arbeitsspeichergröße die Fehlseitenrate exponentiell ansteigt, wobei auch die optimalen Verfahren nicht ausgenommen sind. Da dieser kritische Punkt nicht fest ist, sondern programm- und verfahrensabhängig schwankt und nicht im voraus bekannt ist, muß genügend realer Arbeitsspeicher verfügbar sein, damit das System mit einiger Sicherheit im Bereich akzeptabler Fehlseitenraten arbeitet; als Richtwert gilt $\frac{1}{2}$ bis $\frac{1}{3}$ der virtuellen Programmgröße.

Will man die Fehlseitenrate bei gegebener Arbeitsspeichergröße verringern, so gibt es mehrere Ansatzpunkte. Der erste und - wie schon mehrfach erwähnt - derjenige mit dem größten Verbesserungspotential ist die Verbesserung der Programmlokalität.
Die Programmierer sind hierbei zumindest insofern angesprochen, als sie keinen negativen Einfluß nehmen sollten. Logisch ist der virtuelle Speicher dem realen Speicher äquivalent, vom Zeitverhalten während einer Programmausführung her ist er es jedoch nicht; deshalb sind beim Programmieren gewisse Grundprinzipien, die von MORRISSON /M 4/ bereits 1973 zusammengestellt wurden, zu beachten, um eine gute Performance nicht nur des eigenen Programms, sondern des ganzen Systems sicher-

zustellen. Viel mehr als die Beachtung gewisser Grundprinzipien kann von einem normalen Anwendungsprogrammierer nicht erwartet werden, da er i.a. weder die Interna des Betriebssystems noch die Vorgehensweise des Compilers oder Linkers im einzelnen kennt.

So bleibt an herkömmlichen Methoden zur Verbesserung der Lokalität (oder besser des Verhältnisses Fehlseitenrate/Arbeitsspeichergröße) abgesehen von gewissen Möglichkeiten beim Übersetzungs- und Linkvorgang die Programmrestrukturierung.

Bereits mehrfach erwähnt wurde die Restrukturierung, um die Lokalität von Programmen zu verbessern /H 2, F 3/.
Bei guten Erfolgen sind als Nachteil der hohe Aufwand und die Programmabhängigkeit zu nennen (programmabhängig insofern, als die Restrukturierung für jedes Programm separat durchgeführt werden muß und bereits kleine Programmänderungen den Erfolg in Frage stellen können). Weitergehend hat FERRARI /F2, F 4, F 5/ Programme restrukturiert nicht mit dem Ziel der Verbesserung der Lokalität, sondern mit dem Ziel, das Programmverhalten an eine bestimmte Strategie eines Seitenaustauschverfahrens bzw. an ein bestimmtes Modell des Programmverhaltens anzupassen. Der Erfolg ist im allgemeinen besser als bei der vorgenannten Methode, der Aufwand ist jedoch insofern größer, als für jede Strategie bzw. für jedes Programmverhaltensmodell ein spezielles Restrukturierungsverfahren entwickelt werden muß, woraus überdies eine Systemabhängigkeit entsteht. Der daraus möglicherweise entstehende Nachteil ist nicht, daß Programme oder wenigstens der Restrukturierungsgewinn nicht portabel ist, weil auf der Ebene, auf der Restrukturierung stattfindet, Portabilität i. a. ohnedies nicht gegeben ist, sondern daß der betriebene Aufwand wertlos werden kann, wenn das Seitenaustauschverfahren geändert wird, was im Laufe der Entwicklung eines Betriebssystems keine Seltenheit ist.
In gewisser Weise kann es auch als Nachteil aller Restrukturierungsverfahren gewertet werden, daß sie wegen des Aufwandes bevorzugt bei solchen Programmen zur

Anwendung kommen könnten, die von hochqualifizierten Leuten geschrieben werden, von denen man annehmen muß, daß sie beim Aufbau ihrer Programme auch Performance-Gesichtspunkte zu berücksichtigen in der Lage sind, und daß insgesamt der Anwendungsbereich beschränkt ist.

Was nach wie vor fehlt, ist ein Verfahren, das als permanent aktive Betriebssystemfunktion Verhaltsweisen aktiver Programme erkennt und protokolliert und noch während der Programmausführung verwertet.
Ein prinzipieller Ansatz dazu wäre die Definition eines adaptiven Seitenaustauschverfahrens, das durch solche Informationen gesteuert werden könnte; ein - allerdings wenig erfolgreicher - Versuch in dieser Richtung ist das von THORINGTON und IRWIN vorgeschlagene adaptive Verfahren SIM /T 1/.
Eine zweite Möglichkeit, solche Informationen zu nutzen, ist Prepaging.

Im folgenden wird eine Methode vorgeschlagen, bei der versucht wird, Charakteristika der Seitenreferenzmuster zu erfassen und nachfolgend durch entsprechendes Prepaging zu verwerten. Da das Verfahren permanent aktiv sein soll, darf es nur wenig Overhead erzeugen und muß mit herkömmlichen Mitteln zu implementieren sein. Es kann bereits an dieser Stelle festgestellt werden, daß dieses Verfahren - wie grundsätzlich alle Prepaging Verfahren - nicht in der Lage ist, die Belastung des Sekundärspeichers zu verkleinern, wohl aber potentiell in der Lage ist, die Fehlseitenrate zu reduzieren.

Ein solches Verfahren kann nur als Zusatz zu einem Demand Paging Verfahren realisiert werden, da unter Umständen keine für ein Prepaging Verfahren verwertbare Informationen vorliegen (dies gilt grundsätzlich am Anfang eines erstmalig laufenden Programms) und deshalb nur Demand Paging möglich ist. Um in einfacher Weise wahlweise Prepaging oder Demand Paging betreiben zu können, muß ein Prepaging Verfahren so definiert sein, daß sich Demand Paging als Grenzfall ergibt; dies ist dann der Fall, wenn nur zum Zeitpunkt einer Fehlseitenbedingung das Laden von Seiten

initialisiert werden kann, wobei grundsätzlich die die Fehlseitenbedingung verursachende Seite zu laden ist, darüber hinaus aber in Abhängigkeit vom Vorliegen von im Sinne des Verfahrens verwertbaren Informationen das Laden einer variablen Anzahl weiterer Seiten initialisiert werden kann, und das unterbrochene Programm erst fortgesetzt werden darf, wenn alle Seiten geladen sind. Eine solche Vorgehensweise wird als Demand Prepaging bezeichnet.

3.3.2 Das Folgeseiten - Verfahren

3.3.2.1 Grundsätzliches

Wie bereits ausgeführt kann aufgrund der gegebenen Randbedingungen nur ein sehr unaufwendiges Verfahren zur Informationssammlung und -verwertung in Frage kommen. Vorgeschlagen wird eine Verkettung der sukzessive Fehlseitenbedingungen erzeugenden Seiten; dies ist der Grundgedanke, für den es verschiedene Möglichkeiten der Realisierung gibt, insbesondere auch verschiedene Varianten für eine dynamische Überprüfung und Korrektur der bereits vorhandenen Informationen.

An zusätzlichem Aufwand erforderlich ist Speicherplatz für den Verkettungsvektor, der die Länge n hat, ein Zeiger, der diejenige Seite indentifiziert, die die letzte zurückliegende Fehlseitenbedingung erzeugt hat, und ein Zeitgeber, der nach irgendeinem Kriterium (hier sind mehrere Modelle denkbar) den Verkettungsvorgang abbricht.

Die Motivation für eine solche Vorgehensweise basiert auf dem inzwischen weitgehend akzeptierten Phasen-/Übergangsmodell des Programmverhaltens, wonach an den Phasenübergängen die Seiten der neuen Lokalität durch sukzessive Fehlseitenbedingungen geladen werden. Es ist wünschenswert, eine einmal erkannte Lokalität, sobald sie wieder auftritt, als ganzes oder zumindest in größeren Abschnitten zu laden. Der Verkettungsalgorithmus kann im Sinne dieser Überlegungen wirksam werden: Er kann Seiten, die sukzessive Fehlseitenbedingungen erzeugen, verketten (d. h. logisch verbinden) und diese Information kann dazu benutzt werden, um bei der Annahme, daß

eine so verkettete Lokalität wieder benötigt wird - Anzeichen dafür ist, z. B. eine durch eine zur Verkettung gehörende Seite hervorgerufene Fehlseitenbedingung -, eine oder mehrere Folgeseiten (= nachfolgende Seiten in der Verkettung; daher der Name 'Folgeseiten'-Verfahren) durch Prepaging zu laden.
Im einfachsten Fall des Verfahrens ist durch die Länge einer Verkettung (diese kann zwischen 0 und n - 1 betragen) automatisch festgelegt, ob und in welchem Umfang Prepaging betrieben werden kann.

Ein auf dem Verkettungsprinzip aufbauendes Verfahren kann nur dann wirksam werden, wenn Lokalitäten während der Programmausführung wiederholt auftreten. Bei Programmen, die nur in geringem Umfang solches Verhalten zeigen (dazu gehören Compiler, Optimizer u. ä.), wird man von vornherein kaum Erfolg erwarten können. Nun sind solche Programme in einer wissenschaftlich-technischen Umgebung zwar zahlenmäßig dominant, was den Gesamtverbrauch der Resourcen angeht jedoch von vergleichsweise geringer Bedeutung. Man kann Erfolge also nur bei Programmen mit einem gewissen Maß an Rekursivität auf einer höheren Lokalitätsebene erwarten. Die möglichen Erfolge sind jedoch aus Aufwandsgründen nicht leicht nachzuweisen. Die Erzeugung von Referenzketten durch Simulation des Programmablaufs und in noch weit stärkerem Maße die Simulation der Seitenaustauschverfahren sind so aufwendig, daß man grundsätzlich auf wenige Sekunden realer Programmlaufzeit beschränkt bleibt. Das zur Verfügung stehende System IBM/370 - 168 leistet ca. 3 MIPS, so daß pro Sekunde Programmlaufzeit ca. 4 - 5 Mio. Referenzen generiert werden. Auf jeden Fall müßte bei in obigem Sinne rekursiven Programmen eine verlängerte Simulationszeit (d. h. eine Verlängerung der Referenzkette) zu einer relativen Verbesserung der Ergebnisse führen.

Bei einer vertieften Diskussion des Verkettungsprinzips findet man, daß das Hauptproblem darin besteht, daß Programmlokalitäten i.a. keineswegs seitenweise disjunkt sind, so daß über eine längere Laufzeit eindeutige Verkettungen kaum möglich sind. Der

Fall, daß mehrere verschiedene Seiten die gleiche Folgeseite haben, bereitet keine Schwierigkeiten, da die Informationen nur in Verkettungsrichtung benötigt werden und eine umkehrbar eindeutige Verkettung deshalb nicht erforderlich ist. Anders ist es, wenn eine Seite mehrere mögliche Folgeseiten hat. Konkrete Möglichkeiten der Vorgehensweise in einem solchen Fall werden im nachfolgenden Kapitel behandelt; es kann jedoch grundsätzlich - da Demand Paging als Grenzfall enthalten ist - auf Prepaging verzichtet werden und somit die nächste zu ladende Seite auf Demand Basis ermittelt werden.

Bei der Betrachtung der möglichen Ursachen für die oben beschriebenen Mehrdeutigkeiten ergeben sich zwei Alternativen:

(1) Mehrere verschiedene Programmteile können aufgrund der Programmlogik den gleichen Programmteil als Vorläufer haben.

(2) Die nachfolgenden Programmteile haben logisch verschiedene Vorgänger, die nur zufällig auf der gleichen physikalischen Seite untergebracht sind.

Für die in (2) angesprochenen Fälle kann man davon ausgehen, daß ihre Häufigkeit durch Verkleinerung der Seitengröße reduziert werden kann. Da eine Verkleinerung der Seitengröße jedoch vielfältige Konsequenzen hat, dürfte es schwierig sein, diesen Effekt isoliert nachzuweisen.

Im Fall (1) gibt es kaum Lösungsmöglichkeiten. Zwar wäre es denkbar den negativen Effekt durch Vergrößerung des Verkettungsintervalls zu vermeiden, weil nach MADISON und BATSON /M 1/ die Programmphasen höherer Stufen dazu tendieren, bezüglich ihrer Seiten disjunkt zu sein; dies kann jedoch nicht ohne weiteres als reale Lösungsmöglichkeit akzeptiert werden, weil dadurch die durch Prepaging überdeckten Zeitintervalle sehr groß werden, was zum einen zu einer erheblichen Vergrößerung des bei nichtoptimalen Verfahren stets vorhandenen Prepaging Risikos führt, zum anderen aber auch einen starken Anstieg bei der Belegung realen Arbeitsspeichers zur Folge hat.

3.3.2.2 Erstellung und Überprüfung der Verkettungsinformation

Bei der Erstellung, Überprüfung und Korrektur der Verkettung sind verschiedene Vorgehensweisen möglich.
Es wird ein vernünftig erscheinendes dynamisches Verfahren vorgestellt und begründet, das - mit gewissen verfahrensbedingten Modifikationen - bei allen Simulationen zur Anwendung kommt. Im Anschluß daran werden zwei stark vereinfachte Verkettungsverfahren vorgestellt und die Performance der darauf aufbauenden Folgeseitenverfahren an ausgewählten Beispielen überprüft.

Dynamisches Verfahren

Ziel der Verkettung ist es, die zu einer Lokalität gehörenden Seiten zu verketten. Dazu bedarf es eines Zeitparameters T, der dazu verwendet wird, den Verkettungsvorgang an geeigneter Stelle abzubrechen. Dabei wird entsprechend dem Phasen-/Übergangsmodell vorausgesetzt, daß die zu einer Lokalität gehörenden Seiten innerhalb von T Zeiteinheiten referiert werden und somit auch beim Aufbau der Lokalität innerhalb von T Zeiteinheiten Fehlseitenbedingungen erzeugen, d.h. eine Seite, die mehr als T Zeiteinheiten nach dem Beginn einer Verkettung Page Fault verursacht, wird als nicht zur aktuellen Lokalität gehörend betrachtet, sondern als Beginn einer neuen, und somit wird für die Seite, die den letzten zurückliegenden Page Fault verursacht hat, keine Folgeseite eingetragen. Es ist bemerkenswert, daß der hier definierte Zeitparameter T in seiner Bedeutung identisch ist mit dem Parameter des DWS-Verfahrens, der ja auch genau dasjenige Intervall beschreibt, innerhalb dessen die zu einer Lokalität gehörenden Seiten referiert werden und umgekehrt, daß innerhalb dieser Zeit nicht referierte Seiten nicht zur Lokalität gehören und deshalb auch nicht dem Working Set angehören (d. h. aus dem Arbeitsspeicher zu entfernen sind).
Aufgrund der identischen Bedeutung ist der zunächst unabhängige Verkettungsparameter T bei Kombination des Folgeseiten-Verfahrens mit dem DWS-Verfahren stets größengleich mit der Fenstergröße dieses Verfahrens zu wählen, um ein konsistentes

Gesamtverfahren zu bekommen.

Der Verkettungsparameter kann aber auch in der Logik des PFF-Verfahrens betrieben werden, und wird bei Kombination des Folgeseitenverfahrens mit dem PFF-Verfahren auch in gleicher Logik und größengleich mit dem T-Parameter des PFF-Verfahrens verwendet. Dabei wird davon ausgegangen, daß beim Aufbau einer neuen Lokalität Fehlseitenbedingungen mit relativ hoher Frequenz auftreten, für eine komplett im Arbeitsspeicher befindliche Lokalität eine längere unterbrechungsfreie Laufzeit kennzeichnend ist, so daß der zeitliche Abstand aufeinander folgender Fehlseitenbedingungen als Kriterium herangezogen werden kann; d. h. ein neues Verkettungsintervall wird begonnen und damit das vorhergehende abgeschlossen, wenn der letzte Page Fault mehr als T Zeiteinheiten zurückliegt.

Eine einmal etablierte Verkettung kann nicht als permanent gültig angesehen werden; dies bedeutet, daß einmal eingetragene Verkettungen fortwährend überprüft und eventuell korrigiert werden müssen, selbst dann, wenn nicht die bereits erwähnte grundsätzliche Schwierigkeit nicht eindeutiger Folgeseiten besteht; ein zweiter Grund dafür ist, daß nur die Page Fault verursachenden Seiten verkettet werden, bereits im Arbeitsspeicher befindliche Seiten der gleichen Lokalität aber nicht erfaßt werden können, so daß der Arbeitsspeicherinhalt zu Beginn einer neuen Lokalität Einfluß auf die Verkettung hat. Auch die Verwendung vorhandener Verkettungsinformation durch Prepaging führt - gerade wenn es erfolgreich ist - zu einer Veränderung der Fehlseitenfolge, die in diesem Falle jedoch den gewünschten Verfahrenserfolg anzeigt und nicht zu einer Korrektur der Verkettungsinformation führen sollte, und zwar auch dann nicht, wenn kein maximales Prepaging betrieben wird, d. h. nicht unbedingt alle Folgeseiten einer Fault verursachenden Seite geladen werden, wofür es - wie im nächsten Kapitel begründet wird - gute Gründe geben kann.

Bei echten programmbedingten Mehrdeutigkeiten in einer Verkettung ist die Vorgehensweise wie folgt (Bild 3.9):

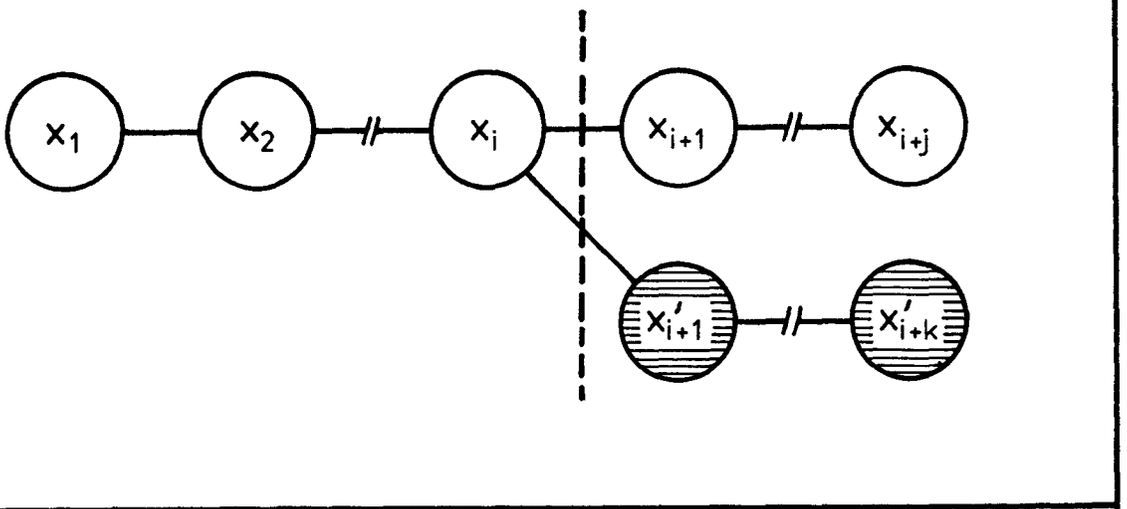


Bild 3.9
Nicht eindeutige Verkettung

Annahme: Es bestehe eine Verkettung der Seiten x_1 bis x_{i+j}. Wenn dann später die Seite $x'_{i+1} \neq x_{i+1}$ nach der Seite x_i Page Fault verursacht, dann wird die bestehende Verkettung aufgetrennt, indem die Folgeseiteninformation für x_i gelöscht wird. Im weiteren Programmverlauf werden dann die Seiten x'_{i+1} bis x'_{i+k} zu einem neuen Zweig verkettet. Bei einem späteren erneuten Laden einer Seite $x_l (l \leq i)$ durch Page Fault würde der Abschnitt der Verkettung bis x_i geladen und dann in Abhängigkeit davon ob entweder die Seite x_{i+1} oder x'_{i+1} den nächsten Page Fault verursacht, die Teilkette x_{i+1} bis x_{i+j} oder x'_{i+1} bis x'_{i+k} geladen; überdies wird die Fault verursachende Seite (d. h. x_{i+1} oder x'_{i+1}) wieder mit x_i verkettet.

Allgemeiner werde die Wahrscheinlichkeit, daß nach der Seite i die Seite j Fehlseitenbedingung verursacht mit q_{ij} bezeichnet; wenn $q_{ij} \neq 0$ für mehr als ein j bei festem i gilt, dann existiert keine eindeutige Verkettung. Eine Möglichkeit, in einem solchen Fall zu einer Verkettung zu kommen, könnte darin bestehen, eine Eintragung in Übereinstimmung mit den bis zu diesem Zeitpunkt gemessenen Übergangshäufigkeiten vorzunehmen. Eine solche Lösung wäre jedoch vom Aufwand her in dem gestecktem Rahmen nicht akzeptabel, da einer Reihe von Seiten eine variabel lange Liste von Übergangszählern zugeordnet werden müßte; sie wäre aber auch nicht dem Programmverhalten angepaßt, da Programme aufgrund von Lokalitätseigenschaften zu bestimmten Zeiten ganz bestimmte Wege bevorzugen, so daß das Aufspalten einer Verkettung und eine anschließende Neuverkettung wie vorher beschrieben angemessener erscheinen.

Beschreibung der Erstellung, Überprüfung und Korrektur von Verkettungen (vgl. Bild 3.10).

Ausgangspunkt ist ein Page Fault, verursacht durch die Seite mit der Nummer SNR; PT1, PTS, PTF enthalten Seitennummern, die als Pointer in der nach Seitennummern geordneten Liste der Verkettungen verwendet werden; PT1 zeigt auf diejenige Seite, die den letzten zurückliegenden Page Fault (vor SNR) verursacht hat.

(1) Im einfachsten Fall ist für die Seite, auf die PT1 zeigt, noch keine Folgeseite eingetragen; in diesem Fall wird die Seite SNR dort (d. h. für die Seite ⟨PT1⟩) als Folgeseite eingetragen, falls die Verkettungszeit nicht überschritten ist, d. h. seit Beginn der Verkettung nicht mehr als eine vorgegebene Anzahl von Zeiteinheiten vergangen ist.

(2) Aufwendiger ist die Prozedur, wenn für die durch PT1 markierte Seite bereits eine Folgeseite eingetragen ist. In diesem Falle muß die von dieser Seite ausgehende Verkettung durchlaufen werden, bis eine von drei möglichen

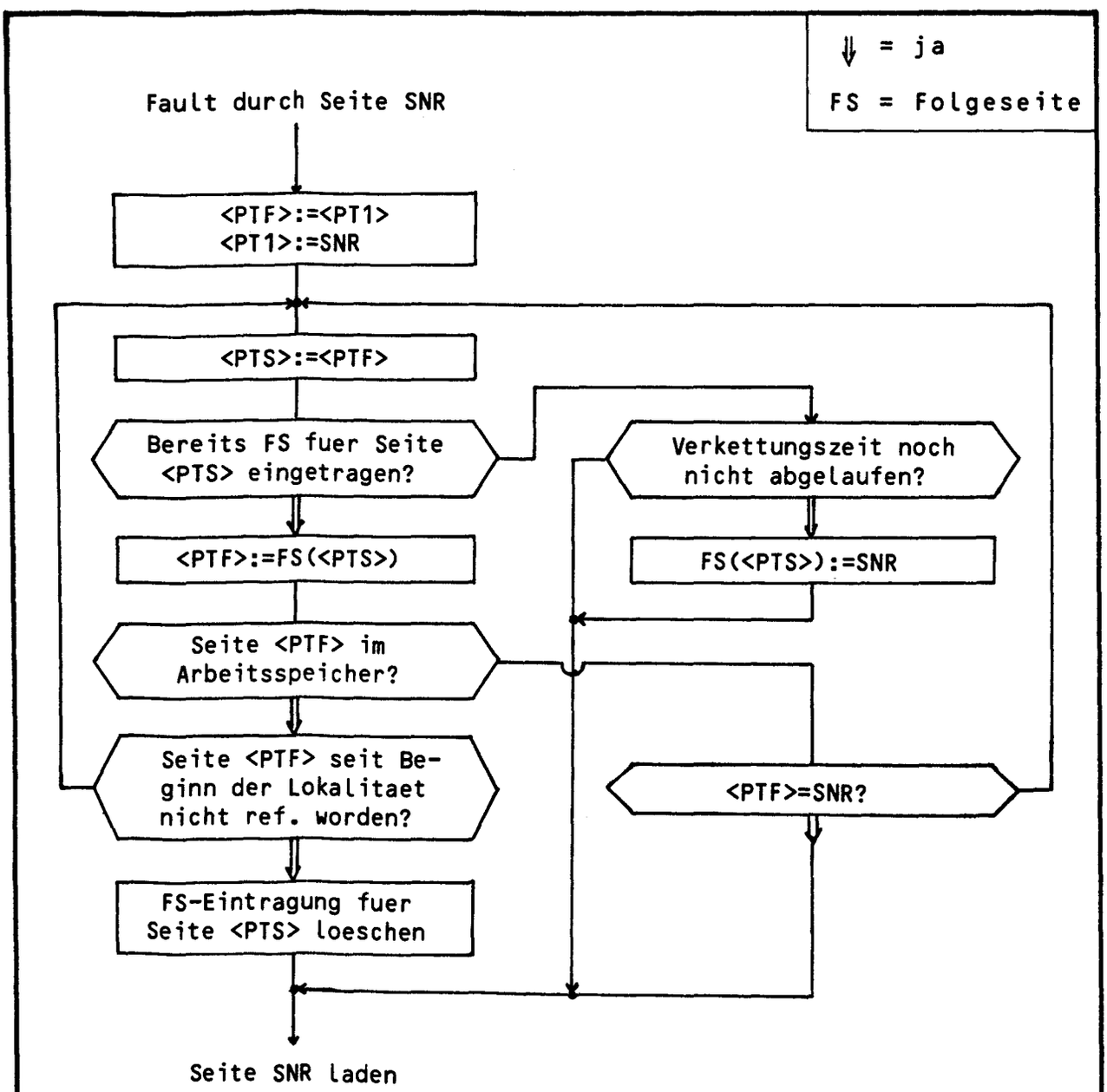


Bild 3.10
Flussdiagramm der Erstellung, Ueberpruefung und Korrektur von Verkettungen

Bedingungen auftritt:

(a) Eine der zur Verkettung gehörenden Seiten befindet sich nicht im Arbeitsspeicher.

Falls dies die Seite SNR ist, ist die bestehende Verkettung in Ordnung, andernfalls wird die bestehende Verkettung an dieser Stelle aufgebrochen durch Löschen der bestehenden Eintragung.

(b) Eine zur Verkettung gehörende im Arbeitsspeicher befindliche Seite wurde seit dem letzten Page Fault nicht mehr referiert; in diesem Falle wird die bestehende Verkettung ebenfalls an dieser Stelle unterbrochen.

(c) Bis zum derzeitigen Ende der Verkettung sind weder Fall (a) noch Fall (b) aufgetreten. In diesem Falle wird die bestehende Verkettung über das bisherige Ende hinaus um die Seite SNR verlängert, falls die Verkettungszeit nicht abgelaufen ist; die Prozedur verläuft analog zu (1).

Die Art und Weise, wie die für die vorgenannten Entscheidungen benötigten Informationen vorliegen und abgefragt werden können, ist von der jeweiligen Ersetzungsstrategie abhängig.

1. Vereinfachtes Verfahren

Bei diesem Verfahren entfällt die Überprüfung bereits vorhandener Verkettungen; es wird jeweils ohne Überprüfung bereits bestehender Verkettungen jeweils die aktuelle Folgeseite eingetragen; die Verkettungen werden wie vorher über einen Zeitparameter beendet (vgl. Bild 3.11).

2. Vereinfachtes Verfahren

Hierbei entfällt neben der Überprüfung der bereits vorhandenen Verkettungen auch noch die zeitgesteuerte Beendigung der Verkettungen und damit der Zeitgeber. Notwendig für ein auf der Basis einer solchen Verkettung arbeitendes Prepaging

Verfahren ist eine verfahrensseitige Beschränkung des Prepaging.

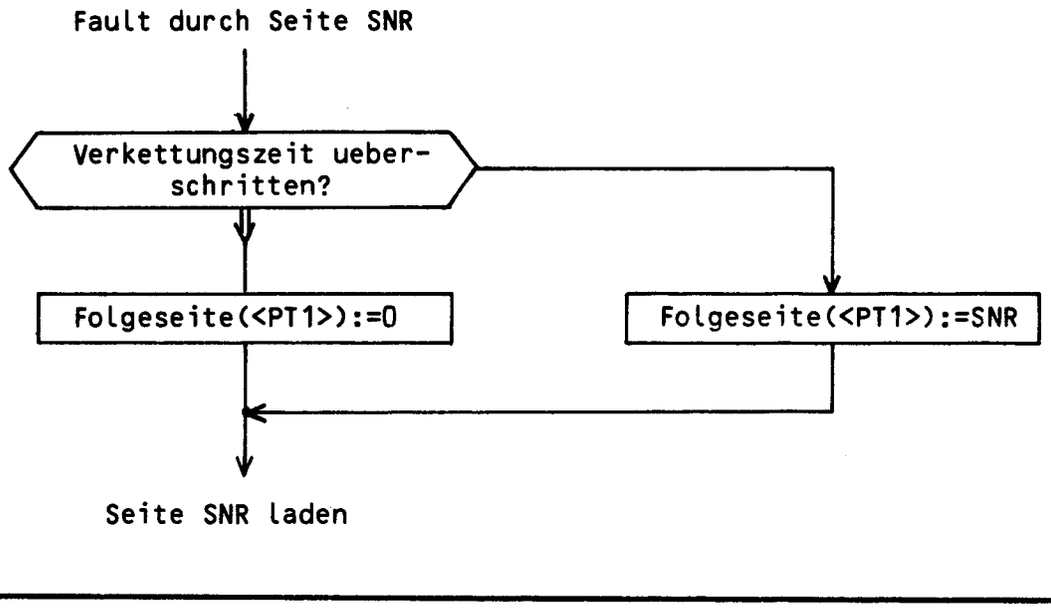


Bild 3.11
Flussdiagramm der Erstellung und Korrektur von Verkettungen nach dem ersten vereinfachten Verfahren

3.3.2.3 Nutzung der Verkettungsinformation

Der Grundgedanke besteht darin, die in Form von Verkettungen vorliegende Information durch Prepaging zu nutzen.

Grundsätzlich wird die Verkettungsinformation bei allen Verfahren in gleicher Weise genutzt; einige Vorgänge - insbesondere die Behandlung durch Prepaging geladener bzw. bereits im Arbeitsspeicher befindlicher Seiten einer Verkettung durch den Replacement Algorithmus - sind verfahrensabhängig und müssen deshalb für jedes Verfahren separat beschrieben werden.

Im folgenden werden einige verfahrensunabhängige Problemstellungen bei der Nutzung der Verkettungsinformation diskutiert und die Lösungen dafür begründet.

Hier ist zunächst die Frage einer zusätzlichen Kontrolle der Maximalzahl der durch Prepaging ladbaren Seiten. Eine natürliche Begrenzung ergibt sich aus der Länge einer Verkettung; es wurde aber bereits darauf hingewiesen, daß eine steuerbare und evtl. schärfere Begrenzung der Zahl der durch Prepaging ladbaren Seiten sinnvoll sein kann, und dies soll im folgenden begründet werde.

Sei $i_o\, i_1\, i_2 \ldots i_l$ $(l \leq n)$ eine Verkettung von Seitennummern.
Die Wahrscheinlichkeit dafür, daß die Seite i_k nach der Seite i_j Page Fault verursacht, sei $q_{i_j i_k}$; die Wahrscheinlichkeiten der obigen Verkettung sind $q_{i_o i_1}$, $q_{i_1 i_2}$,..., $q_{i_{l-1} i_l}$.

Für alle Wahrscheinlichkeiten gilt $0 \leq q_{i_j i_k} \leq 1$, wobei 0 bei einer existierenden Verkettung nicht auftreten kann, während für $q_{i_j j_k} = 1$ eine eindeutige Zuordnung zu einer bestimmten Lokalität notwendig ist; i. a. gilt $q_{i_j j_k} < 1$, wobei es - lokales Programmverhalten vorausgesetzt - stark bevorzugte Übergänge gibt.
Die Wahrscheinlichkeit, daß die obige Verkettung als Ganzes auftritt bzw. erneut auftritt, ist

$$Q_{i_o i_1 \ldots i_l} = \prod_{k=1}^{l} q_{i_{k-1} i_k},$$

nimmt also wegen $q_{jk} \leq 1$ mit der Länge der Verkettung ab. Somit ist es klar, daß es sinnvoll sein kann, das Ausmaß, mit dem Prepaging betrieben wird, zu begrenzen, um das Risiko, nicht benötigte Seiten zu laden, klein zu halten. Konkret wird dies durch Vorgabe eines Parameters PREPLIM realisiert, der die maximale Anzahl der pro Fault ladbaren Seiten angibt. Es zeigt sich bei den Messungen, daß bei unstrukturierten Programmen, die, wie bereits vorher dargelegt, ohnedies nur bescheidene Ergebnisse erwarten lassen, auch die Gefahr des Ladens nicht benötigter Seiten am größten und damit die Begrenzung dieses Risikos durch Vorgabe kleiner Werte für PREPLIM am dringendsten ist.

Ein weiterer wichtiger Bestandteil aller Verfahren ist die Schleifenkontrolle. Es ist grundsätzlich nicht auszuschließen, daß Seiten zu einer geschlossenen Schleife verkettet werden können. Dies kann ausgeschlossen werden bei bestimmten Verfahren, wenn der Zeitparameter der Verkettung und der Parameter des Ersetzungsverfahrens aufeinander abgestimmt sind, z. B. beim DWS-Verfahren, wenn die Verkettungszeit in gleicher Logik betrieben und zahlenmäßig kleiner oder gleich der Fenstergröße des DWS-Verfahrens ist.

Da Schleifenbildung sehr selten auftritt, ist es nicht notwendig, eine Lösung zu implementieren, die ein sofortiges Erkennen einer Schleife gestattet; eine solche Lösung wäre aufwendig, da bei jeder neuen Folgeseite zu prüfen wäre, ob sie von allen vorhergehenden Seiten der Verkettung verschieden ist. Es genügt eine Kontrolle durch Überwachung der Maximallänge eine Verkettung, die durch die Anzahl der Programmseiten (n) gegeben ist. Da bis zum Erkennen einer Schleife nach diesem Kriterium diese bereits mehrfach durchlaufen worden sein kann, ist das Programm so anzulegen, daß mehrfaches Auftreten gleicher Seiten in einer Verkettung keinen Schaden anrichtet.

3.3.2.4 Zusatzinformation der Simulation

Bei der Simulation der Prepaging-Verfahren werden einige Größen mitgeführt, die für die Durchführung der Verfahren nicht notwendig sind, die aber geeignet sind, die sich abspielenden Vorgänge zu erhellen.
Es werden ermittelt und ausgegeben

(1) für einen Simulationslauf:

   (a) die Anzahl der abgeschlossenen Lokalitäten,

   (b) die Anzahl der durch Prepaging geladenen Seiten, die unreferiert wieder aus dem Arbeitsspeicher entfernt wurden.

(2) Für jeden Page Fault:

   (a) die Gesamtlänge der mit der Fault verursachenden Seite beginnenden

Verkettung,

(b) welche und wieviele Seiten dieser Verkettung geladen wurden,

(c) wieviele Seiten der Verkettung sich am Ende des Ladevorganges im Arbeitsspeicher befanden.

Mit Hilfe dieser Informationen können die Vorgänge sehr detailliert rekonstrukturiert werden. Die unter (1 b) genannte Anzahl der unreferiert aus dem Arbeitsspeicher entfernten Seiten, ist ein Maß für die Qualität des Prepaging Verfahrens, denn dies sind genau die unnötigerweise geladenen Seiten. Diese Zahl ist jedoch im Zusammenhang mit dem Verfahrensgewinn (Größe der Reduktion der Fehlseitenrate) und dem eingegangenen Verfahrensrisiko , das über die Vorgabe von PREPLIM steuerbar ist, zu sehen.

3.3.2.5 Spezielle Verfahren

Bei allen Verfahren wurde eine der bekannten Ersetzungsstrategien mit einer Prepaging Ladestrategie auf der Basis des Folgeseitenalgorithmus kombiniert. Hierbei wurde grundsätzlich versucht, das ursprüngliche Verfahren in seiner Substanz so wenig wie möglich zu verändern; dies führt bezüglich der Nutzung der Verkettungsinformation zu einer Adaption an das Basisverfahren, die nachfolgend beschrieben und begründet wird.

(a) LRU

Das LRU-Verfahren ist ein Verfahren fester Arbeitsspeichergröße, für das eine Prioritätsliste existiert, in der die Seiten nach der Dauer ihres unreferierten Aufenthalts im Arbeitsspeicher geordnet sind. Zunächst stellt sich die Frage, ob durch Prepaging zu ladende Seiten zusätzlich zur vorgegebenen Speichergröße m geladen werden sollen oder ob das Prepaging innerhalb der vorgegebenen Speichergröße vor sich gehen soll. Wenn die vorzeitig geladenen Seiten zusätzlich geladen würden, ginge ein

wesentliches Merkmal des LRU-Verfahrens verloren: es entstände dann nämlich ein Verfahren variabler Arbeitsspeichergröße. Zusätzlich würde auch der Performance-Vergleich zwischen dem Basisverfahren und dem Prepaging Verfahren erschwert, da man korrekterweise das Prepaging Verfahren mit einem Basisverfahren vergleichen müßte, dessen feste Arbeitsspeichergröße gleich der mittleren Arbeitsspeichergröße des variablen Verfahrens ist. Auch aus praktischen Gründen scheint es nicht sinnvoll zu sein, den Mehraufwand, den die Implementation eines größenvariablen Verfahrens systemseitig erfordert, hinzunehmen für ein Verfahren, das in seiner Grundsubstanz ein Verfahren fester Arbeitsspeichergröße und deshalb wohl nicht geeignet ist, den Mehraufwand optimal zu nutzen. Es wird deshalb Prepaging innerhalb der vorgegebenen Arbeitsspeichergröße m betrieben.

Eine zweite Frage ist, welche Position im Stack und damit in der Prioritätsliste die durch Prepaging geladenen oder die bereits im Arbeitsspeicher befindlichen Seiten einer Verkettung haben sollen.

Unbestritten - und typisches Charakteristikum eines Stack - ist, daß die zuletzt referierte Seite - und das ist zu einem Ladezeitpunkt stets die Fault verursachende Seite - die oberste Stack-Position einnehmen muß. Da es sich um ein Verfahren fester Arbeitsspeichergröße handelt, muß nach der Aufbauphase auch für jede Seite, die vorzeitig geladen werden soll, eine Seite aus dem Arbeitsspeicher entfernt werden; dabei müssen jedoch Seiten der gleichen Verkettung, die sich zufällig bereits im Arbeitsspeicher befinden, ausgenommen werden.
Da es bezüglich der Verkettung unerheblich ist, ob eine Seite einer Verkettung sich bereits im Arbeitsspeicher befindet oder durch Prepaging geladen werden muß, ist es sinnvoll, alle Seiten einer Verkettung bezüglich ihrer Positionierung im Stack gleich zu behandeln, d. h., nicht zwischen geladenen und bereits im Speicher befindlichen Seiten zu unterscheiden. Sei die Länge einer Verkettung k ($k \leq m$) einschließlich der Fault verursachenden Seite, dann bieten sich zwei Lösungen an:

Das Fault verursachende Startelement der Verkettung bekommt - wie bereits erläutert - grundsätzlich die oberste Stackposition zugeordnet; die Positionen 2 bis k der Verkettung werden

(a) den Stack-Positionen 2 bis k zugeordnet,

(b) den Stack-Positionen m - k + 2 bis m zugeordnet.

Der signifikante Unterschied der beiden Alternativen liegt in der Behandlung der Seiten durch den Replacement - Algorithmus bei nachfolgenden Fehlseitenbedingungen; im Falle (b) sind die Seiten der Verkettung die ersten, die aus dem Arbeitsspeicher entfernt werden, wohingegen sie im Falle (a) möglichst lange im Arbeitsspeicher gehalten werden.

Bei der Simulation wurde gemäß (a) verfahren, und zwar aus folgenden Gründen:

(1) Wenn die Seiten der Verkettung tatsächlich wie erwartet referiert werden bevor weitere Fehlseitenbedingungen auftreten, dann unterscheiden sich Fall (a) und (b) in ihren Auswirkungen nicht, da die referierten Seiten dann in jedem Fall an die Spitze des Stack rücken und von da an in gleicher Weise dem Alterungsprozeß des LRU-Verfahrens unterliegen.

(2) Wenn die Seiten einer Lokalität nicht in gleicher Sequenz referiert werden wie bei der Erstellung der Verkettung und die Verkettung nicht vollständig geladen wurde (z. B. durch Wirksamwerden des Parameters PREPLIM), dann können durchaus weitere Fehlseitenbedingungen auftreten, um die durch die Verkettung möglicherweise nur unvollständig beschriebene Lokalität zu vervollständigen; im Falle (b) könnten dabei vorher geladene Seiten dieser Lokalität schon wieder aus dem Arbeitsspeicher entfernt werden.

(3) Wenn eine eindeutige Zuordnung von Seiten zu Lokalitäten besteht, wird i. a. der unter (1) allenfalls noch der unter (2) beschriebene Sachverhalt zutreffen. Wenn eine solche eindeutige Zuordnung jedoch nicht gegeben ist, dann besteht die Möglichkeit, daß Seiten geladen werden, die im Rahmen der gerade aufzubauen-

den Lokalität nicht benötigt werden. In diesem Falle besteht - wie auch Messungen bestätigt haben - eine realistische Chance, daß von diesen im Zusammenhang mit der akatuellen Lokalität zwar nicht benötigten Seiten einige im weiteren Programmverlauf doch noch referiert werden (und damit nicht unnötig geladen wurden), wenn sie ausreichend lange im Arbeitsspeicher bleiben.

(b) DWS

Das DWS-Verfahren ist ein Verfahren variabler Arbeitsspeichergröße, das dadurch gekennzeichnet ist, daß Seiten, die innerhalb der letzten zurückliegenden T Zeiteinheiten (virtuell) referiert wurden, zum Working Set gehören und der Working Set eines Programms komplett im Arbeitsspeicher sein muß, wenn das Programm sich in Ausführung befindet. Diese Bedingung ist dominant, so daß das DWS-Verfahren über diese Bedingung in die Klasse der Verfahren mit impliziter Steuerung des Multiprogramming-Grades gehört.
Aus diesem Grunde werden beim DWS-Verfahren die duch Prepaging zu ladenden Seiten zusätzlich zum Working Set, der durch Vorgabe des Zeitparameters T definiert ist, geladen.

Es soll an dieser Stelle erwähnt werden, daß Prepaging innerhalb des vorgegebenen Working Set zu einem interessanten Alternativverfahren führt, das sich jedoch stärker vom Basis-DWS-Verfahren entfernt. Bei einer solchen Vorgehensweise müssen zum Working Set gehörende Seiten aus dem Arbeitsspeicher entfernt werden; sinnvollerweise werden dafür solche Seiten gewählt, die nach dem DWS-Algorithmus als nächste entfernt worden wären (d. h. diejenigen, die am längsten nicht mehr referiert worden sind); dies entspricht einer Verkleinerung von T. Ein solches Prepaging Verfahren führt also zu einem variablen DWS-Parameter T, wobei der nur zu Prepaging Zeitpunkten auftretende Effekt dem einer Verkleinerung von T entspricht. Legt man nun wieder das Phasen-/Übergangsmodell zugrunde und unterstellt, daß Prepaging vorwiegend beim Aufbau neuer Lokalitäten an Phasenübergängen wirksam wird, dann ist der Effekt einer

Verkleinerung von T an diesen Stellen sehr erwünscht, weil er dem bereits mehrfach erwähnten kurzfristigen Aufblähen des Working Set beim DWS-Verfahren entgegenwirkt. Diese kurzzeitige Expansion, die zustande kommt, weil an Phasenübergängen Teile beider benachbarten Lokalitäten zum Working Set des DWS-Verfahrens gehören, würde gemildert, weil die durch Prepaging geladenen Seiten auf Kosten von Seiten der alten Lokalität in den Arbeitsspeicher gelangen.

Dadurch, daß bei dem durchgeführten Verfahren die vorzeitig geladenen Seiten zusätzlich zu dem durch das Originalverfahren definierten Working Set geladen werden, ergibt sich beim Prepaging Verfahren grundsätzlich eine höhere mittlere Arbeitsspeicherbelegung für den gleichen Wert des DWS-Parameters. Dies braucht bei Performance-Vergleichen jedoch nicht explizit berücksichtigt zu werden, da bei den üblichen Performance-Darstellungen das gewählte Performance-Maß über der mittleren (bzw. der fest vorgegebenen) Arbeitsspeichergröße aufgetragen wird.
Die Frage der Behandlung der zu einer Verkettung gehörenden Seiten läßt sich beim DWS-Verfahren in einer naheliegenden, verfahrenskonformen Weise lösen:

(1) Die durch Prepaging geladenen Seiten gelten als zum Ladezeitpunkt (quasi-)referiert und unterliegen von diesem Zeitpunkt an dem üblichen Alterungsprozeß und damit dem Ersetzungsmechanismus des Verfahrens; dies ist sinnvoll, da beim DWS-Verfahren davon ausgegangen wird, daß eine Seite, die zur aktuellen Lokalität gehört, innerhalb von T Zeiteinheiten referiert wird.

(2) Es besteht keine logische Differenz zwischen solchen Seiten einer Verkettung, die durch Prepaging geladen werden und den Seiten, die sich (zufällig) bereits im Arbeitsspeicher befinden; sie sind deshalb auch bezüglich ihrer Bewertung im Replacement-Algorithmus gleich zu behandeln, d. h., die bereits im Arbeitsspeicher befindlichen Seiten gelten als im gleichen Moment (quasi-)referiert wie die vorzeitig geladenen Seiten. Dies ist ein Eingriff in das ursprüngliche Verfahren, der zu dem vorgenannten Argument der Gleichbehandlung auch deshalb gerecht-

fertigt ist, weil es nicht sinnvoll wäre, eine Seite, von der erwartet wird, daß sie in naher Zukunft referiert wird, möglicherweise unmittelbar vorher zu einem zufälligen Zeitpunkt aus dem Arbeitsspeicher zu entfernen. Durch diese Vorgehensweise erhält jede sich im Arbeitsspeicher befindliche Seite einer Verkettung in gleicher Weise für ein definiertes Zeitintervall der Länge T die Chance, referiert zu werden.

(c) PFF

Das PFF-Verfahren ist wie das DWS-Verfahren ein Verfahren, bei dem der aktuelle Working Set ausschließlich aufgrund der Programmeigenschaften (d. h. unabhängig von der Umgebung) geschätzt wird, so daß die vorrangig geforderte Bereitstellung des kompletten Working Set im Arbeitsspeicher zu einer impliziten Steuerung des Multiprogramming-Grades führt.
Es werden deshalb auch hier die durch Prepaging zu ladenden Seiten zusätzlich zum aktuellen Working Set des Originalverfahrens geladen.

Da beim PFF-Verfahren ein Page Fault Zeitpunkt potentiell auch stets Zeitpunkt einer Überprüfung des Arbeitsspeicherinhaltes ist (eine Überprüfung findet genau dann statt, wenn der letzte Page Fault mehr und die letzte Überprüfung nicht weniger als T Zeiteinheiten zurückliegt), sind Prepaging Vorgang und Überprüfungsaktion so anzuordnen, daß im Arbeitsspeicher befindliche Seiten einer Verkettung nicht unmittelbar vor dem Ladevorgang aus dem Arbeitsspeicher entfernt werden.

In der vorliegenden Realisierung des Verfahrens läuft die logische Operation 'Ladevorgang' vor der Arbeitsspeicherüberprüfung ab. Dabei werden alle zu der aktuellen Verkettung gehörenden Seiten, das sind die Fault verursachende Seite, die durch Prepaging zu ladenden Seiten und die bereits im Arbeitsspeicher befindlichen Seiten als 'seit dem letzten Page Fault referiert' markiert, so daß sie von einer unmittelbar anschließenden Arbeitsspeicherüberprüfung nicht betroffen wären. Bei dieser Anordnung werden die zur Verkettung gehörenden Seiten - in Übereinstimmung

mit der Definition des PFF-Verfahrens - bei der nächstfolgenden Arbeitsspeicherüberprüfung aus dem Speicher entfernt, falls sie in der Zwischenzeit nicht wirklich referiert wurden. Es bleiben bei einer Arbeitsspeicherüberprüfung also diejenigen Seiten im Arbeitsspeicher, die nach dem letzten Ladevorgang referiert wurden.

Die Replacement-Algorithmen halten diejenigen Seiten im Arbeitsspeicher, die nach dem das Verfahren kennzeichnenden Kriterium die höchste Wahrscheinlichkeit besitzen, in naher Zukunft wieder referiert zu werden. In analoger Weise etablieren auch die Folgeseiten-Verfahren eine Art Prioritätsliste, da erwartet wird, daß die Seiten einer Verkettung - mit abnehmender Wahrscheinlichkeit - in naher Zukunft referiert werden, wenn die Ausgangsseite referiert wurde. Bei den Folgeseiten-Verfahren werden die im Sinne der Verkettung höchstwertigen Seiten entweder wie beim DWS- und PFF-Verfahren zusätzlich zum Working Set des Basisverfahrens oder wie beim LRU-Verfahren anstelle der geringstwertigen Seiten des Working Set des Basis-Verfahrens geladen. Daß diese Vorgehensweise nicht à priori unsinnig ist, beweisen Messungen mit PREPLIM = 1, wo - da die Fault verursachende Seite mitzählt - überhaupt kein Prepaging betrieben wird, sondern nur bei den zufällig im Arbeitsspeicher vorhandenen Seiten, die oben beschriebene Anhebung der Priorität im Sinne des Basis-Verfahrens stattfindet, und bereits dies i. a. zu einem deutlichen Rückgang der Anzahl der Page Faults führt.

3.3.2.6 Aufwandsbetrachtungen

Beim Prepaging nach den Folgeseiten-Verfahren entsteht an zwei Stellen zusätzlicher Aufwand gegenüber den zugrunde liegenden Demand Paging Verfahren:

(a) bei der Erstellung der Verkettung und

(b) bei der Nutzung der Verkettungsinformation durch Prepaging.

Einen Eindruck vom erforderlichen Aufwand für die Erstellung der Normalverkettung vermittelt das Flußdiagramm in Bild 3.10. Der dadurch entstehende Overhead ist nicht

gering, er läßt sich jedoch nur schwer abschätzen, da zum Verkettungsvorgang Suchvorgänge gehören, deren Dauer vom Verkettungszustand, d.h. sowohl vom Programm als auch von den Parametervorgaben abhängt. Die in Kap. 4.3.5 präsentierten Untersuchungsergebnisse zeigen jedoch, daß auf der Basis des vereinfachten Verkettungsalgorithmus V1 praktisch gleichwertige Folgeseiten-Verfahren entstehen. Der Verkettungsalgorithmus V1 ist in Bild 3.11 dargestellt und erzeugt einen nur sehr geringen Overhead in der Größe weniger Maschinenbefehle, der mit 20 bis 30 μs pro Verkettungsvorgang (d.h. pro Fehlseitenbedingung) abgeschätzt werden kann. Der sich daraus ergebende Gesamt-Overhead zur Laufzeit hängt von der Anzahl der Fehlseitenbedingungen ab und würde bei 200 Page Faults pro Sekunde, was bei heutigen Systemen bereits ein hoher Wert ist, etwa 0,5 % betragen.

Was den Aufwand für das Prepaging angeht, so bestehen Unterschiede zwischen den Verfahren DWS+ und PFF+, bei denen die durch Prepaging geladenen Seiten zum bestehenden Working Set hinzukommen, und dem Verfahren LRU+, bei dem das Prepaging innerhalb der fest vorgegebenen Arbeitsspeichergröße stattfindet. Der Overhead, der bei den erstgenannten Verfahren durch vorzeitiges Laden einer Seite entsteht, ist gering und durch die folgenden Operationen gekennzeichnet:

(1) Abfrage: Ist eine Folgeseite zur aktuellen Seite eingetragen?

(2) Falls ja, Abfrage: Befindet sich diese Folgeseite bereits im Arbeitsspeicher?

(3) In Abhängigkeit vom Abfrageergebnis (2):

  (a) Falls die Seite sich im Arbeitsspeicher befindet: Seite so kennzeichnen, als ob sie geladen worden wäre (Anhebung der Priorität, Pseudo-Referenzbit setzen o. ä.).

  (b) Wie bei einer Page Fault verursachenden Seite Transfer vorbereiten.

(4) Loop-Kontrolle durch einfaches Zählen der Seiten einer Verkettung.

Alle vorgenannten Operationen liegen auf dem Niveau von Maschineninstruktionen; insbesondere beinhalten die Abfragen keine Suchoperationen, da die abzufragenden

Informationen in entsprechend geordneten Listen vorliegen.

Es kann somit festgestellt werden, daß die Vorbereitung des Page-I/O für eine durch Prepaging zu ladende Seite nur geringfügig aufwendiger ist als bei einer Page Fault verursachenden Seite. Gleichzeitig muß darauf hingewiesen werden, daß die Durchführung des Page-I/O bei den Prepaging Verfahren durchweg weniger Overhead produziert, da im allgemeinen mehrere Seiten auf einmal geladen werden.

Da das normale PFF-Verfahren als Sonderfall des Folgeseiten-Verfahrens simuliert wurde, existieren vergleichbare Simulationszeiten für die beiden Verfahren. Um das PFF-Verfahren zu simulieren, wurde der Programmteil 'Erstellung der Verkettungsinformation' beim Verfahren PFF+ weggelassen, wodurch Prepaging entfällt, da keine verwertbaren Verkettungsinformationen vorhanden sind. Die Simulationszeiten für das Verfahren PFF+ enthalten sowohl den Aufwand für die Erstellung der Verkettung (und zwar für die aufwendigere Normalverkettung) als auch für die Durchführung des Prepaging als auch die Zeiten, die für die Erstellung der in Kap. 3.3.2.4 beschriebenen Zusatzinformationen anfallen. Die Simulationszeiten (Summe der CPU-Zeiten über alle Parameterwerte) sind in Tab. 3.1 zusammengestellt.

Programm	$\sum$ CPU-Zeit (Minuten)		
	PFF	PFF+ PREPLIM=2	PFF+ PREPLIM=10
P1	40,22	39,80	39,96
P2	120,14	118,80	118,97
P3	108,18	107,25	107,42
P4	143,69	142,37	142,10
P5	47,47	46,88	46,67

Tab. 3.1
Simulationszeiten für die Verfahren PFF und PFF+

In allen Fällen sind die Simulationszeiten für das Verfahren PFF+ kürzer als für das normale PFF-Verfahren. Dieses zunächst erstaunliche Ergebnis bedarf einer Diskussion.

Zunächst ist festzuhalten, daß sich die CPU-Zeit-Angaben nicht auf eine Implementation, sondern auf die Simulation der Verfahren beziehen. Dies bedeutet zum einen, daß darin keine Zeiten für die Durchführung des Page-I/O enthalten sind, zum anderen, daß darin Zeiten für die Abarbeitung jeder einzelnen Referenz enthalten sind, die bei einer Implementation nicht anfallen würden. Die Zeiten setzen sich wie folgt zusammen:

$$\sum \text{CPU-Zeit (PFF)} = A + B$$

$$\sum \text{CPU-Zeit (PFF+)} = A + B',$$

wobei $A = \sum_{i=1}^{|\omega|} \text{CPU-Zeit (Ref. i)}$ den Anteil beschreibt, der für die Abarbeitung aller Referenzen anfällt und der trotz des relativ kleinen Beitrags pro Referenz wegen der großen Anzahl der Summanden dominieren wird, wohingegen B bzw. B' den Overhead beschreibt, der durch Page Faults bzw. durch Page Faults und Prepaging entsteht. Wegen der Dominanz des Summanden A muß davon ausgegangen werden, daß Unterschiede zwischen B und B' sich auf die Gesamtsumme nur relativ gering auswirken. Dies wird durch die Zahlen in Tab. 3.1 bestätigt; da jedoch in allen Fällen die Zeiten für das Verfahren PFF+ unter denen des PFF-Verfahrens liegen, kann dies trotz der relativ geringen Differenzen von etwa 1 % kein zufälliges Ergebnis aufgrund von Ungenauigkeiten oder Umgebungsabhängigkeiten der CPU-Zeit-Messung sein. Die Frage ist, wie dies zustande kommen kann, da doch nach dem vorher Gesagten beim Verketten und bei der Nutzung der Verkettungsinformation zusätzlicher Overhead gegenüber dem normalen PFF-Verfahren entsteht. Die Erklärung ist, daß der Overhead für die Vorbereitung des Page-I/O nicht der einzige verfahrensbedingte Overhead ist; insbesondere kommen beim PFF-Verfahren die relativ aufwendigen Überprüfungen der Arbeitsspeicherinhalte hinzu, deren Häufigkeit wesentlich durch die Anzahl der Fehlseitenbedingungen (nicht der Ladevorgänge) bestimmt ist, die beim Prepaging

Verfahren deutlich niedriger ist als beim normalen PFF-Verfahren. Offensichtlich ist es so, daß der Mehraufwand bei den Prepaging Verfahren, der beim Page-I/O unzweifelhaft vorhanden ist, durch Einsparungen aufgrund der verringerten Zahl von Fehlseitenbedingungen mehr als aufgewogen werden kann.

Der Prepaging-Algorithmus des Verfahrens LRU+ erzeugt einen größeren Overhead als der der Verfahren DWS+ und PFF+. Der Overhead, der durch die Vorbereitung der Page In's entsteht, ist durch die gleichen Operationen gekennzeichnet wie bei den vorgenannten Verfahren und ähnlich gering. Beim Verfahren LRU+ kommt jedoch hinzu, daß nach der Aufbauphase für jede Seite, die durch Prepaging geladen werden soll, eine Seite aus dem Arbeitsspeicher entfernt werden muß, da das Prepaging innerhalb der Speicherplatzvorgabe abläuft. Dies führt nicht grundsätzlich zu einer Benachteiligung gegenüber dem normalen LRU-Verfahren, da auch für jede auf Demand-Basis zu ladende Seite eine andere Seite aus dem Arbeitsspeicher entfernt werden muß; die Ersetzungsstrategie ist beim Prepaging Verfahren jedoch aufwendiger als beim Normalverfahren, da es nicht sinnvoll ist, eine Seite der aktuellen Lokalität (gemäß Verkettungsinformation) aus dem Arbeitsspeicher zu entfernen, um eine andere Seite der gleichen Lokalität laden zu können. Um sicherzustellen, daß dies nicht geschehen kann, muß für jede Seite, die gemäß Ersetzungsstrategie auszuladen wäre, überprüft werden, ob sie nicht zur aktuellen Lokalität gehört, und gegebenenfalls ist eine solche Seite zu blockieren und der Ersetzungsmechanismus erneut anzuwenden. Im Grenzfall, wenn keine nicht zur aktuellen Lokalität gehörende Seite im Arbeitsspeicher ist, führt dies zu einer Abbruchbedingung für das Prepaging. Verkompliziert wird die Ersetzungsstrategie noch dadurch, daß an dieser Stelle eine Unterscheidung notwendig ist zwischen Seiten, die auf Demand Basis bzw. durch Prepaging geladen werden sollen, da erstere auch dann geladen werden müssen, wenn dies nur auf Kosten einer anderen Seite der gleichen Lokalität möglich ist. Das Wirksamwerden der vorgenannten Abbruchbedingung bedeutet zwar Maximierung des Overheads, erspart u. U. aber überflüssige I/O-Vorgänge und kann überdies nur auftreten, wenn eine

Lokalität nicht in den Arbeitsspeicher paßt, was als Thrashing-Kondition anzusehen ist; in diesem Sinne kann diese Abbruchbedingung als Indikator (und zwar ein a priori anzeigender) für eine Thrashing-Kondition ausgewertet werden.

Wie groß der bei der Ersetzungsstrategie entstehende zusätzliche Overhead ist, hängt von den Verkettungslängen und von der Wahrscheinlichkeit ab, mit der die Ersetzungsstrategie mehrfach anzuwenden ist. Beides ist sowohl von Parametervorgaben für das Verfahren als auch von Programmeigenschaften abhängig.

Um auch in diesem Fall zumindest Anhaltspunkte für eine quantitative Erfassung der Overheads zu bekommen, wurden an Programm P1 (vgl. Kap. 4.1) CPU-Zeit-Messungen der Verfahren LRU und LRU+ durchgeführt. Dabei wurde das normale LRU-Verfahren nicht nach der in Kap. 2.6.1 beschriebenen vereinfachten Methode, sondern als Grenzfall des Verfahrens LRU+ simuliert; beim Verfahren LRU+ wurde die Verkettung gemäß der vereinfachten Variante V1 durchgeführt und auf die Erzeugung der in Kap. 3.3.2.4 beschriebenen Zusatzinformationen verzichtet. Die Simulationen der Verfahren LRU und LRU+ wurden jeweils unmittelbar hintereinander ausgeführt, um eine vergleichbare Maschinenumgebung zu gewährleisten; die Meßergebnisse sind in Tab. 3.2 zusammengestellt.

Verfahren	$\sum$ CPU-Zeit (Min)	$\sum$ Page Faults	$\sum$ Page Loads
LRU	36,87	16895	16895
LRU+ PREPLIM=2	37,16	7005	9752
LRU	37,01	16895	16895
LRU+ PREPLIM=10	37,00	6196	12072

Tab. 3.2
P1: Simulationszeiten für die Verfahren LRU und LRU+

Die Meßwerte zeigen in der Tendenz das gleiche Ergebnis wie beim PFF-Verfahren: Je Page Fault ist der Overhead beim Prepaging Verfahren merkbar größer als beim Normalverfahren; bei gutem Verfahrenserfolg ist jedoch durch Reduktion der Anzahl der Page Faults der Gesamtoverhead des Prepaging Verfahrens dem des Normalverfahrens vergleichbar. Bei den Aussagen ist zu berücksichtigen, daß die Simulation der Verfahren so geschieht, daß nur zu Page Fault Zeitpunkten der LRU-Stack so weit konstruiert wird, daß die Ersetzungsstrategie anwendbar ist, und damit ein erheblicher Teil des Verfahrens-Overheads zu diesen Zeitpunkten anfällt; der Gesamt-Overhead ist deshalb in relativ starkem Maße von der Anzahl der Page Faults abhängig.

4 MESSUNGEN - ERGEBNISSE

4.1 Allgemeines

Bei der Bewertung der Ergebnisse ist zu berücksichtigen, daß die Seitenaustauschverfahren - gleich, ob Demand-oder Prepaging Verfahren - für sehr kleine Arbeitsspeicher ($m, \overline{m} \rightarrow 1$) und für sehr große Arbeitsspeicher ($m, \overline{m} \rightarrow n$) ihre Identität verlieren und die Ergebnisse fast ausschließlich vom zufälligen Programmverhalten abhängig sind. Was hierunter zu verstehen ist, wird nachstehend am Beispiel der Grenzwerte verdeutlicht.

Für $m = 1$ (dem entspricht $T = 1$ bei den Verfahren variabler Arbeitsspeichergröße) führt jede Referenz einer von der vorhergehenden verschiedenen Seite verfahrensunabhängig zu einer Fehlseitenbedingung, d. h., Anzahl und Zeitpunkte der Fehlseitenbedingungen sind ausschließlich von der Referenzkette ω (d. h. durch die Individualität des Programmlaufs) bestimmt.

Ähnlich ist die Situation für $m \geq n$ (dem entspricht $T \geq |\omega|$): Es treten nur die initialen Page Faults auf, deren Anzahl und Zeitpunkte wiederum nur vom Programm abhängen; das Folgeseitenverfahren z. B. reduziert sich auf das darin enthaltene Demand Verfahren, da Prepaging auf der Basis von während der Programmausführung ermittelten Programmeigenschaften nicht möglich ist. Bei optimalen Prepaging Verfahren verschwindet auch der Programmeinfluß weitgehend: am Anfang der Programmausführung werden alle n Seiten des Programmes geladen.

Man muß also davon ausgehen, daß die Messungen in den Randbereichen nur geringe Aussagekraft haben; unter praktischen Gesichtspunkten ist dies kein gravierender Nachteil, da den Randbereichen in der Praxis nur geringe Bedeutung zukommt.

Die Messungen wurden an fünf verschiedenen Programmen durchgeführt. In Tab. 4.1 sind die wichtigsten Daten zur Kennzeichnung der Programme festgehalten.

Programm	Größe in Seiten zu		Länge der Referenzkette
	2048 Bytes	256 Bytes	
P1: Erzeugung und Inversion einer 50 x 50 Matrix	90	586	3 716 361
P2: Dreimalige Erzeugung und Inversion einer 50 x 50 Matrix	93	590	11 156 814
P3: Eigenwertberechnung einer 50 x 50 Matrix	40	193	10 131 455
P4: Berechnung der Ackermann-Funktion für die Werte (3,5)	67	407	13 337 324
P5: Übersetzung eines PL1-Programms mit dem PL1-Optimizer	139	618	4 218 934

Tab. 4.1

Simulationsprogramme

Die Programme P1 bis P4 sind Fortran-Programme, P1 bis P3 entstammen der IMSL (International Mathematical and Statistical Library), sind also Bausteine für größere Anwenderprogramme. Für die Erstellung der Referenzketten wurde ein an der Universität Dortmund entwickeltes Simulationsprogramm verwendet. Dieses Programm liefert pro Referenz die Seitennummer der referierten Adresse, bezogen auf eine Seitengröße von 256 Bytes, wobei die Nummern so angeordnet sind, daß durch einfache Division Seitennummern für Seiten bis zu 4 k Bytes Größe erzeugt werden können; ferner wird ein C-Bit (Change Bit: gesetzt, falls Schreibzugriff zum Speicher)

und ein I-Bit (Instruction Bit: gesetzt für Instruktionszugriffe) erzeugt.

Für die Simulation folgender Seitenaustauschverfahren wurden Programme entwickelt:

LRU, OPT, LRU+, PREOPT, OPR;

DWS, VMIN, DWS+, VMINPP1, VMINPP2;

PFF, PFF+ .

Dabei bezeichnet XXX+ das Prepaging Verfahren auf der Basis des Verfahrens XXX.
Die Simulationsprogramme sind in PL/1 geschrieben. Der Aufwand für die Verfahrenssimulation ist verhältnismäßig gering für die Verfahren LRU, DWS, VMIN, VMINPP1 und VMINPP2, die nach den in Kapitel 2.6 erläuterten vereinfachten Methoden durchgeführt wurden; bei allen anderen Simulationen ist der Aufwand mit etwa einer Minute CPU-Zeit pro eine Million Referenzen und Parameterwert auf einem System IBM/370-168 ziemlich hoch.
Als Standard-Parameter wurden für das DWS- und PFF-Verfahren und die jeweiligen davon abgeleiteten Verfahren die in Tab. 4.2 angegebenen Werte verwendet.
Die Zeiten in Tab. 4.2 sind in Referenzen angegeben.
Bei den Verfahren fester Speichergröße wurden die Vorgaben der Speichergrößen an den Programmgrößen orientiert und nach der Erzeugung der Fehlseitenraten für das LRU-Verfahren für alle $m \leq n$ nach dem in Kap. 2.6 beschriebenen summarischen Verfahren unter Berücksichtigung der Programmeigenschaften festgelegt; Tab. 4.3 enthält die Zusammenstellung.
Die Speichergrößen m (bzw. $\overline{m}$) werden grundsätzlich in Seiten zu 2048 Bytes angegeben. Eine Ausnahme davon wird nur in Kapitel 4.3.4 gemacht, wo es um den Einfluß der Seitengröße geht; dort werden die Angaben in KBytes gemacht.

Die in den folgenden Kapiteln präsentierten und diskutierten Ergebnisse befassen sich schwerpunktartig mit der relativen Performance von Demand Paging Verfahren und Prepaging Verfahren.

	DWS	PFF	
I	T(I)	T(I)	T'(I)
1	1 000	250	250 000
2	5 000	500	250 000
3	7 500	2 500	250 000
4	10 000	4 000	250 000
5	15 000	7 500	250 000
6	25 000	10 000	250 000
7	50 000	15 000	250 000
8	75 000	25 000	250 000
9	125 000	35 000	250 000
10	250 000	50 000	250 000
11	750 000	125 000	250 000
12		250 000	250 000

Tab. 4.2

Standard-Parameterwerte beim DWS- und PFF-Verfahren

I	m(I) in Seiten				
	P 1	P 2	P 3	P 4	P 5
1	20	20	10	10	20
2	30	30	12	15	25
3	35	35	15	20	27
4	40	40	18	25	30
5	45	45	20	30	35
6	50	50	23	35	40
7	55	55	25	40	45
8	60	60	30	45	50
9	70	70	35	50	55
10	80	80	40	55	60
11	90	90	-	65	70

Tab. 4.3

Standard-Vorgaben der Arbeitsspeichergrößen bei Verfahren fester Speichergröße

4.2 Vergleichende Messungen bei optimalen Verfahren

4.2.1 Verfahren fester Arbeitsspeichergröße

Zum Performance-Vergleich der optimalen Verfahren fester Arbeitsspeichergröße ist die Fehlseitenrate der Verfahren OPT, PREOPT und OPR für die Programme P1 bis P5 in den Bildern 4.1 bis 4.5 aufgetragen. Um eine Relation zu den nichtoptimalen Verfahren herzustellen, wurde die Fehlseitenrate des LRU-Verfahrens hinzugefügt; ferner ist die Page In-Rate des OPR-Verfahrens eingezeichnet.

Wie es nach den theoretischen Aussagen in Kap. 3.2.1 sein muß, ist die Reihenfolge (abnehmende Performance) in allen Fällen OPR - PREOPT - OPT - LRU . Auffällig ist in P4 die Gruppenbildung der mit deutlichem Abstand besseren Prepaging Verfahren auf der einen Seite und der Demand Verfahren auf der anderen Seite, wobei innerhalb der Gruppen nur minimale Performance-Unterschiede vorhanden sind, die oben angegebene Reihenfolge jedoch gewahrt bleibt. In allen Fällen verläuft die Page In-Rate des OPR-Verfahrens unterhalb derjenigen des LRU-Verfahrens und oberhalb derjenigen des OPR-Verfahrens; man beachte, daß diese Kurve nicht unterhalb der OPT-Kurve verlaufen kann, da die Anzahl der Ladevorgänge (=Anzahl der Fehlseitenbedingungen) beim OPT-Verfahren minimal ist.
Sehr bemerkenswert ist, daß sich an keiner Stelle und in keinem der Programme an der eindeutigen Performance-Relation OPR - PREOPT etwas ändert, wenn an Stelle der Fehlseitenrate die modifizierte Fehlseitenrate als Performance-Maß gewählt wird (vgl. Tab. 4.4 bis 4.8). Dies bedeutet, daß beim OPR-Verfahren der Performance-Gewinn durch Reduktion der Anzahl der Fehlseitenbedingungen immer größer ist als der Verlust durch zusätzliche Ladevorgänge.
Die gesamten Performance-Daten der Verfahren fester Speichergröße sind in den Tabellen 4.4 bis 4.8 in Anhang A zusammengestellt.

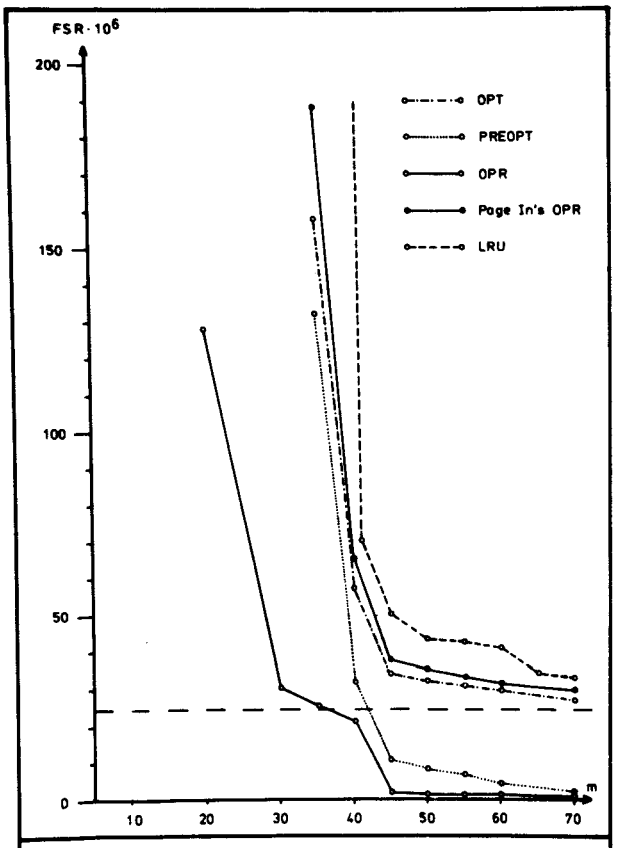


Bild 4.1
P1: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens

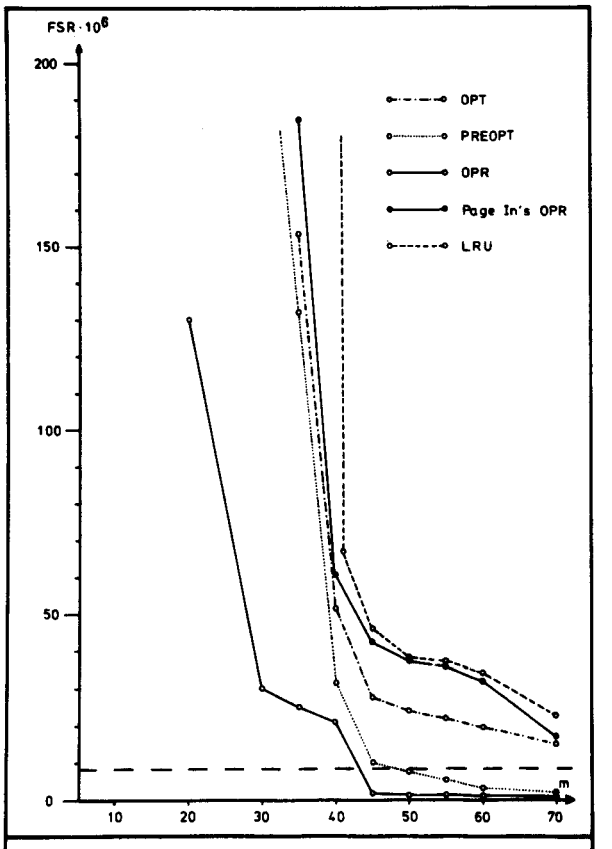


Bild 4.2
P2: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens

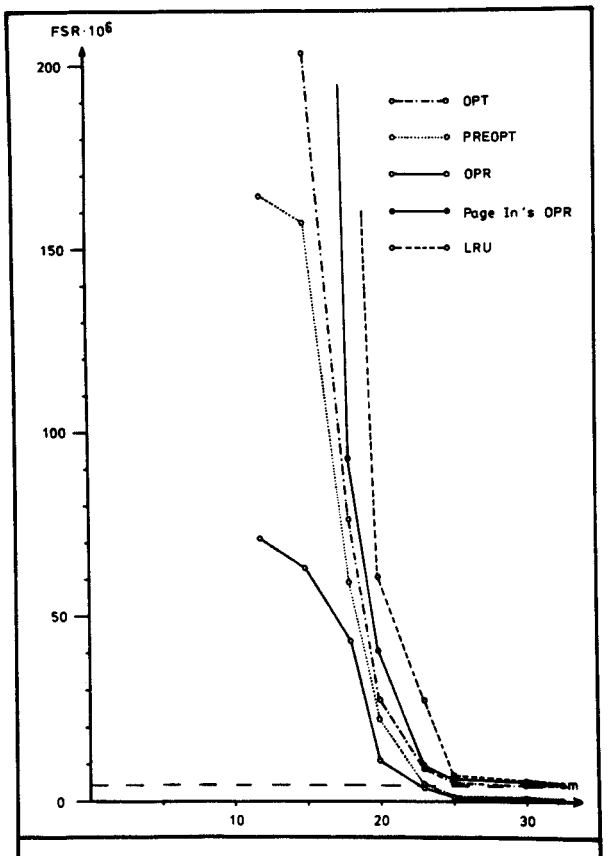


Bild 4.3
P3: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens

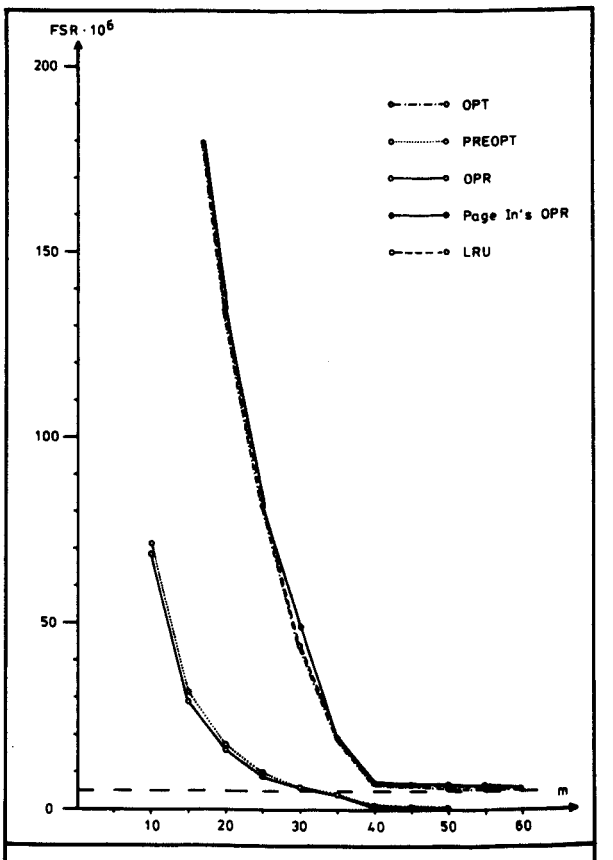


Bild 4.4
P4: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens

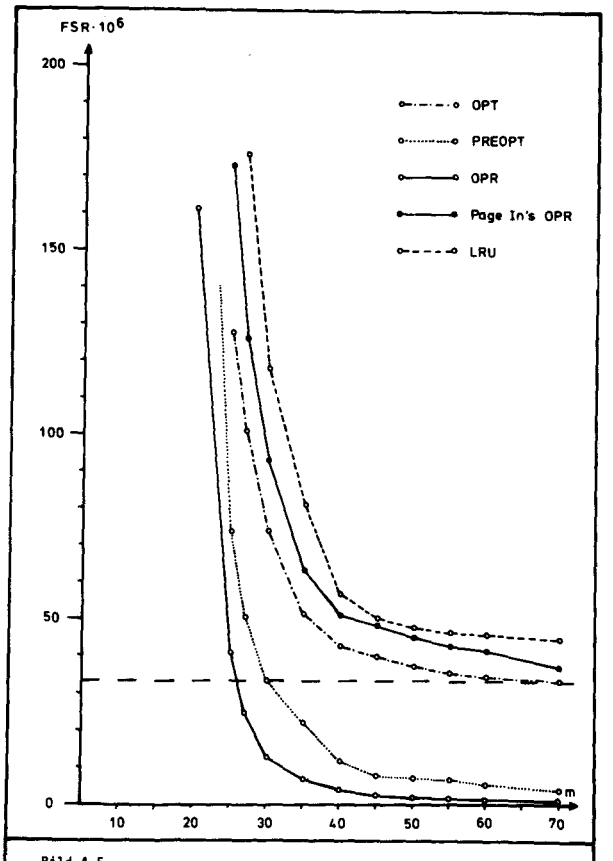


Bild 4.5
P5: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens

4.2.2 Verfahren variabler Arbeitsspeichergröße

Hierbei geht es um den Vergleich des optimalen Demand Paging Verfahrens VMIN mit den Prepaging Verfahren VMINPP1 und VMINPP2, deren Fehlseitenraten über der mittleren Speicherbelegung in den Bildern 4.6 bis 4.10 für die Programme P1 bis P5 aufgetragen sind. Zum Vergleich ist die Fehlseitenrate des nicht optimalen DWS-Verahrens eingezeichnet.

Für das VMINPP1-Verfahren wurde der Verfahrensparameter T' gleich der Fenstergröße T des zugrunde liegenden VMIN-Verfahrens gewählt; beim VMINPP2-Verfahren ist $T' = \frac{T}{2}$.

Die Messungen zeigen das erwartete Ergebnis: Aufgrund der Definition der Verfahren VMINPP1 und VMINPP2 kann bei diesen die Fehlseitenrate nicht größer sein als beim VMIN-Verfahren; bei normalen Programmen liegt sie - wie die Messungen bestätigen -

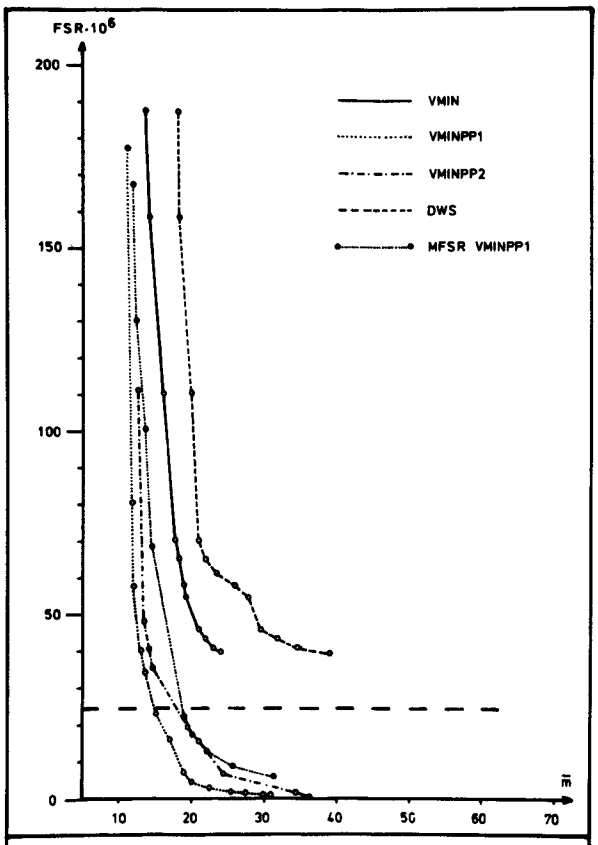


Bild 4.6
P1: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens

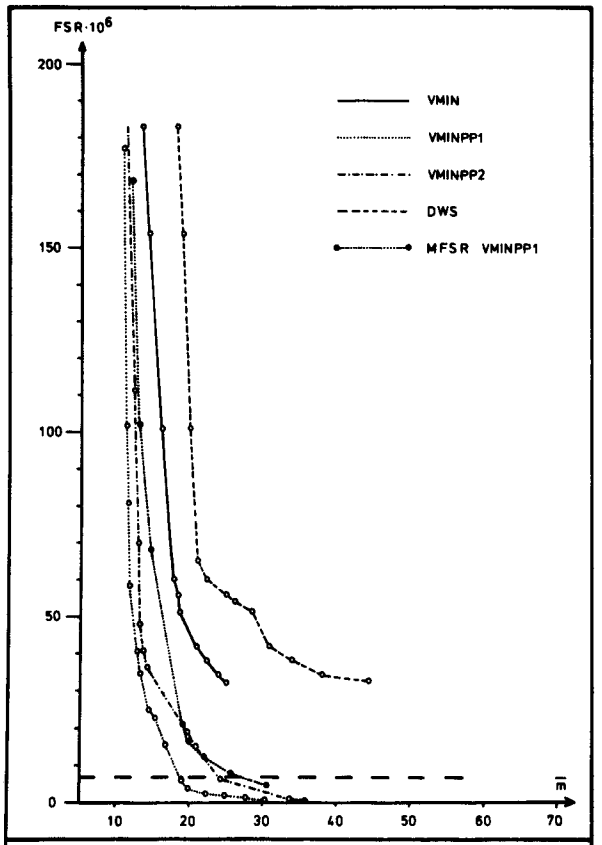


Bild 4.7
P2: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens

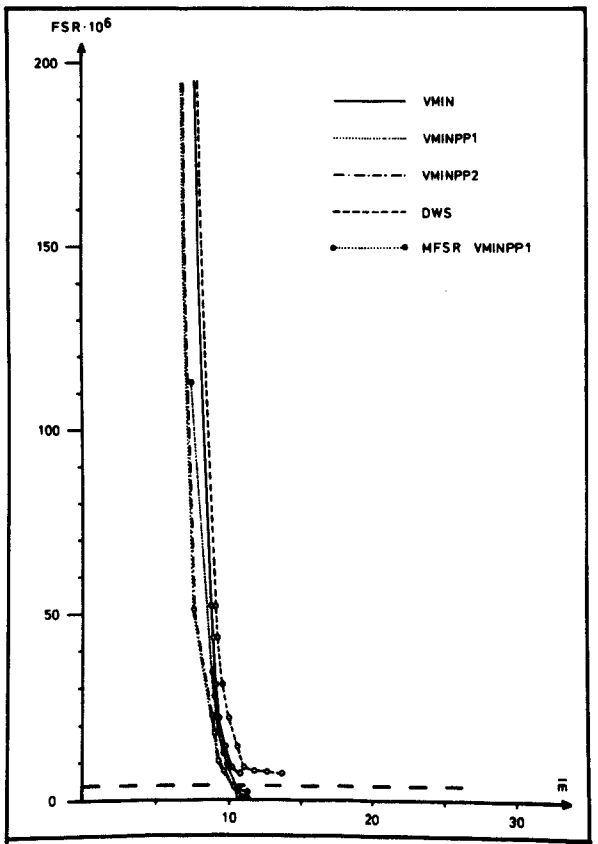


Bild 4.8:
P3: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens

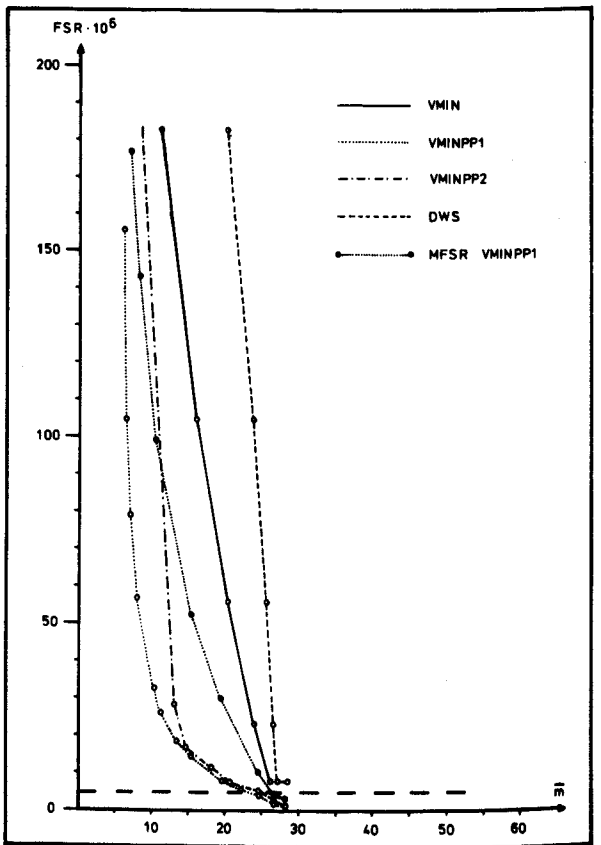


Bild 4.9
P4: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens

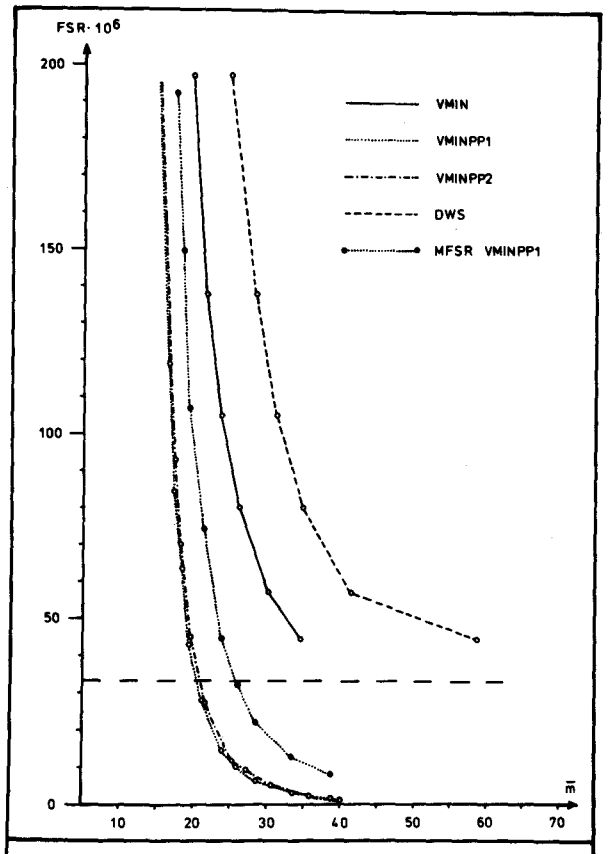


Bild 4.10
Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens

deutlich darunter. Daß die Kurven der Prepaging Verfahren in den Bildern 4.6 bis 4.10 stets unterhalb derjenigen des VMIN-Verfahrens verlaufen, ist jedoch nicht von vornherein klar: zwar produzieren die Prepaging Verfahren definitionsgemäß eine geringere Zahl von Page Faults, gleichzeitig wird durch das vorzeitige Laden von Seiten jedoch die mittlere Speicherbelegung gegenüber dem Basisverfahren VMIN erhöht. Der Verlauf der Kurven in den Bildern zeigt, daß diese Erhöhung weniger gravierend ist als die Reduktion der Fehlseitenrate.

Noch weniger trivial sind die Performance-Relationen, wenn an Stelle der Fehlseitenrate die modifizierte Fehlseitenrate als Performance-Maß benutzt wird.

Beim VMINPP1-Verfahren ist wegen T' = T die Anzahl der Page In's identisch mit der Anzahl der Page Faults (und damit Page In's) des VMIN-Verfahrens, so daß die modifizierte Fehlseitenrate genau das geometrische Mittel der Fehlseitenraten beider Verfahren liefert.

Beim VMINPP2-Verfahren ist die Länge der durch Prepaging überdeckten Zeitabschnitte nicht vorhersagbar und kann insbesondere größer werden als die Fenstergröße T des VMIN-Verfahrens, wodurch die Zahl der Page In's kleiner werden kann als die Zahl der Page Faults des VMIN-Verfahrens (wobei die relative Vergrößerung der mittleren Speicherbelegung mit der Ausdehnung der Prepaging Intervalle wächst). Aus diesem Grunde schneidet das VMINPP2-Verfahren relativ zum VMINPP1-Verfahren etwas besser ab bei Verwendung der modifizierten Fehlseitenrate an Stelle der

Fehlseitenrate.

Für das VMINPP1-Verfahren ist die Performance-Kurve für MFSR in die Bilder 4.6 bis 4.10 eingetragen. Es ist zu sehen, daß auch bei der Verwendung des ungünstigeren Performance-Maßes MFSR das Prepaging Verfahren bei allen gemessenen Programmen und allen Parameterwerten besser abschneidet als das VMIN-Verfahren.

Die Performance-Daten, die in den Tabellen 4.9 bis 4.13 (Anhang A) zusammengestellt sind, zeigen, daß das VMINPP2-Verfahren bei gleichem T des zugrunde liegenden VMIN-Verfahrens meist eine geringere Zahl von Fehlseitenbedingungen bei einer höheren mittleren Speicherbelegung erzeugt verglichen mit dem VMINPP1-Verfahren. Die höhere Speicherbelegung ist dafür verantwortlich, daß - wie es die Bilder zeigen - die Performance des VMINPP2-Verfahrens durchweg etwas schlechter ist als die des VMINPP1-Verfahrens.

Wie in Kap. 3.2.2 angekündigt, wurden auch an den Varianten der Prepaging Verfahren, die sich durch eine abweichende Bewertung der Ladezugriffe ergeben (vgl. Kap. 3.2.2.2), Messungen durchgeführt.

Im Bereich großer und mittlerer T-Werte ergab sich bei beiden Prepaging Verfahren praktisch kein Unterschied zwischen den jeweiligen Varianten; nur bei extrem kleinen T-Werten waren signifikante Unterschiede feststellbar. Die Differenzen entsprachen in der Tendenz den Erwartungen: das Alternativ-Verfahren lieferte weniger Page Faults bei einem weiteren Anwachsen der mittleren Speicherbelegung.

4.3 Vergleichende Messungen bei nichtoptimalen Verfahren

4.3.1 Grundauswertung

Wie in Kapitel 3.3.2.5 im einzelnen beschrieben, wurden Prepaging Verfahren auf der Basis der Demand Paging Verfahren LRU, DWS und PFF formuliert. Bei der Grundauswertung ist die Fehlseitenrate des Basisverfahrens und des Prepaging Verfahrens eingetragen, letztere einmal für unlimitiertes Prepaging (d. h. PREPLIM = n) und für auf zwei Ladevorgänge pro Page Fault beschränktes Prepaging (PREPLIM = 2). Beim LRU-Verfahren ist unlimitiertes Prepaging nicht zulässig, da wegen des begrenzten Arbeitsspeichers stets PREPLIM ≤ m erfüllt sein muß. Um den Prepaging Vorgang von der Vorgabe der Arbeitsspeichergröße zu entkoppeln, wurde beim LRU-Verfahren einheitlich mit PREPLIM = 10 gearbeitet, weil 10 Seiten die kleinste Vorgabe für die Speichergröße ist. Bei den Messungen hat sich diese Einschränkung als nicht gravierend herausgestellt, da die Verkettungen bzw. der Speicherzustand stärkeres Prepaging kaum zulassen.

In den Bildern 4.11 bis 4.15 sind die Ergebnisse für das LRU-Verfahren, in 4.16 bis 4.20 für das DWS-Verfahren und in 4.21 bis 4.25 für das PFF-Verfahren jeweils für die fünf Testprogramme dargestellt. Bei der Darstellung wurde der Bereich der Fehlseitenrate auf 200 Faults pro $1o^6$ Referenzen begrenzt, um eine logarithmische Verzerrung zu vermeiden und dennoch eine ausreichende Auflösung zu haben. Dieser Maximalwert liegt mit deutlichem Sicherheitsabstand über dem, was mit der heutigen Hardware praktisch möglich ist. Das für die Simulationen verwendete System IBM/370-168 mit zweiTrommeln an einem Kanal leistet maximal ca. 60 Transfers pro 10^6 Referenzen, was etwa 300 Übertragungen pro Sekunde entspricht; zu berücksichtigen ist darüber hinaus, daß zusätzlich zu den Page In's auch die Page Out's den Sekundärspeicher belasten und diese einen relativ hohen Anteil der Page In's ausmachen (bei den fünf Testprogrammen durchweg deutlich über 50 %).

Zu den Ergebnissen im einzelnen:

(a) LRU, LRU+

Das LRU-Verfahren liefert im Gegensatz zu den Verfahren variabler Arbeitsspeichergröße keine Anhaltspunkte für eine sinnvolle und verfahrenskonforme Wahl der Verkettungszeiten. Es wurden keine systematischen Meßreihen zur Optimierung der Verkettungszeiten durchgeführt; durch einige Testmessungen wurde jedoch festgestellt, daß es nicht günstig ist, mit einem mittelgroßen festen Wert der Verkettungszeit für alle Speichergrößen zu arbeiten, sondern Vorteile bringt, die Verkettungszeit mit der Vorgabe für die Speichergröße wachsen zu lassen. Bei allen LRU+-Verfahren wurden deshalb ohne weitergehende Untersuchungen die Standardwerte des DWS-Verfahrens indexweise den Speichergrößen zugeordnet.

Bei den Programmen P1 bis P4 ist bemerkenswert, daß sich die verschiedenen Kurven an keiner Stelle schneiden, d. h., es existieren eindeutige Performance-Relationen zugunsten der Prepaging Verfahren, wobei die Verfahren mit nicht limitiertem Prepaging nochmals besser abschneiden als die mit eingeschränktem Prepaging.

Das Programm P5 wurde als Beispiel für ein nicht rekursives Programm gewählt und nimmt als solches erwartungsgemäß eine Sonderstellung ein *. Die Messungen bestätigen, was in Kap. 3.3.2 bereits aufgrund der Überlegungen zur Wirkungsweise des Folgeseitenverfahrens gesagt wurde. Bei nicht rekursiven Programmen ist durch Prepaging auf der Basis von Verkettungsinformationen kaum ein Performance-Gewinn zu erzielen. Bei beschränktem Prepaging tritt zumindest keine Verschlechterung der Performance des Verfahrens LRU+ gegenüber dem

---

* P5 kann als extremes Beispiel gelten, da es die Übersetzung eines fehlerfreien Programms beinhaltet, der Übersetzungsvorgang somit 'straight forward' abläuft, was sich auch an der geringen Anzahl der angesprochenen Seiten (bezogen auf die Programmgröße) ablesen läßt.

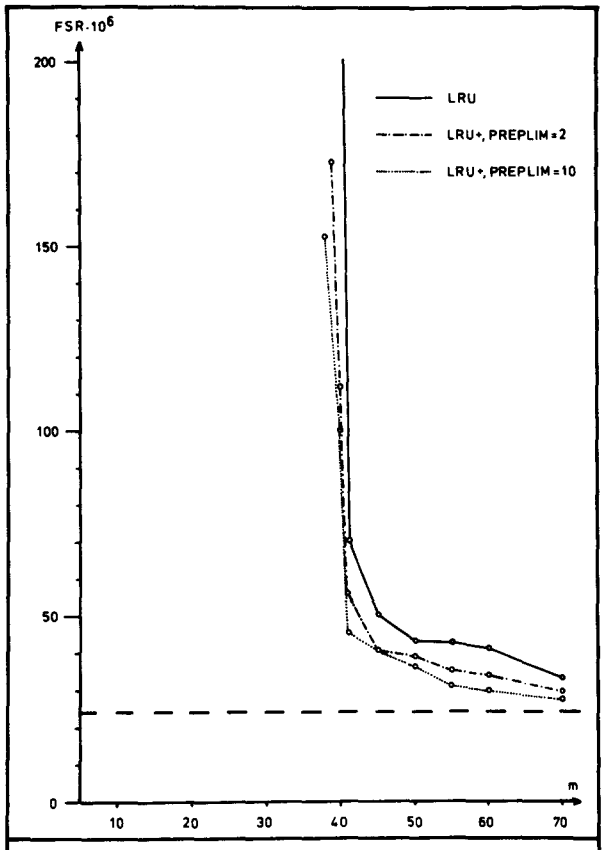


Bild 4.11
P1: Performance der Verfahren LRU und LRU+

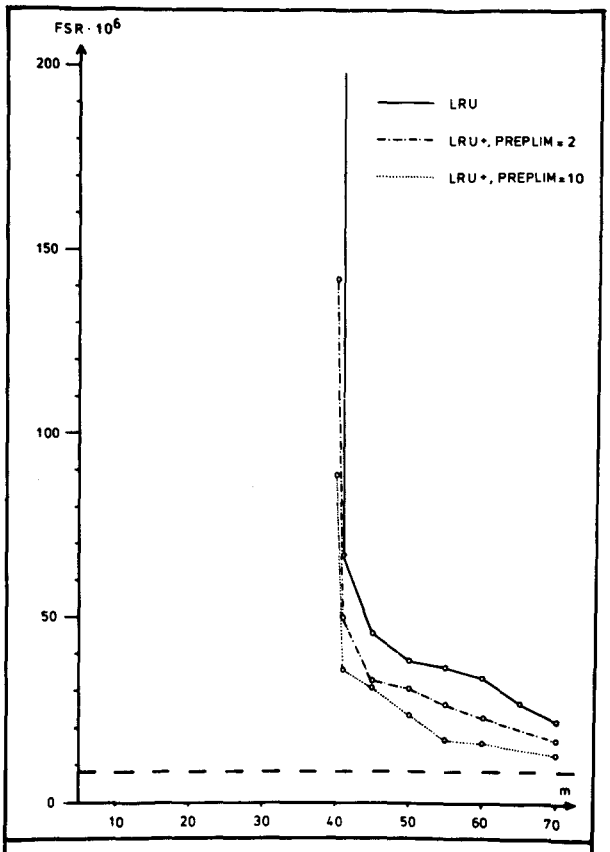


Bild 4.12
P2: Performance der Verfahren LRU und LRU+

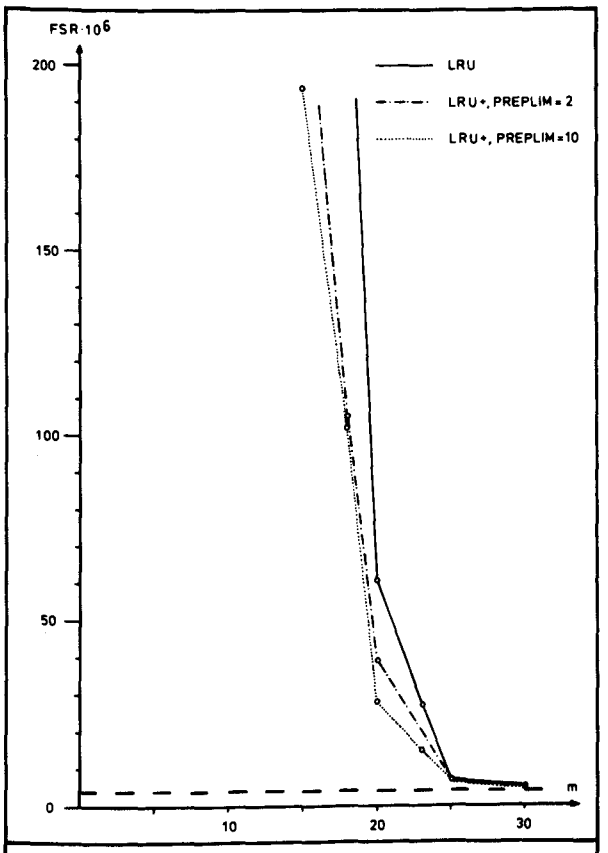


Bild 4.13
P3: Performance der Verfahren LRU und LRU+

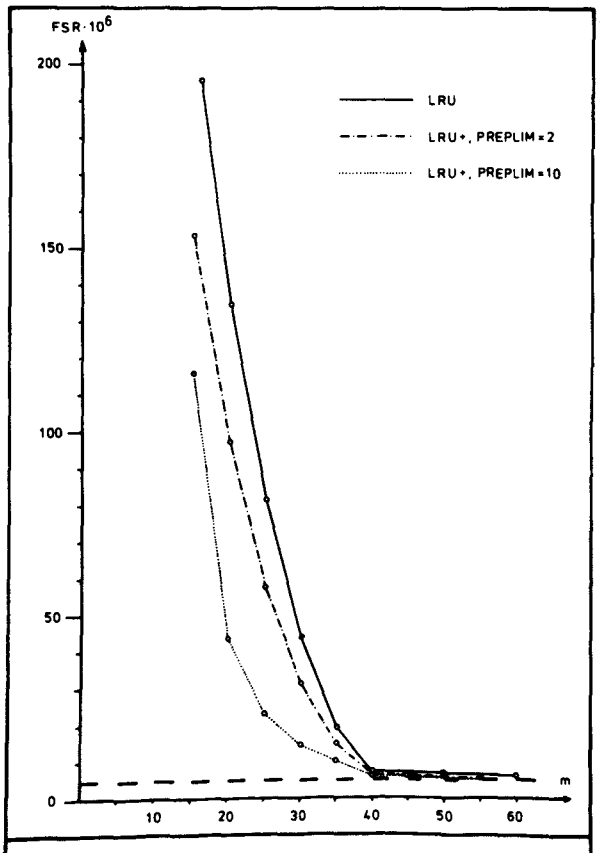


Bild 4.14
P4: Performance der Verfahren LRU und LRU+

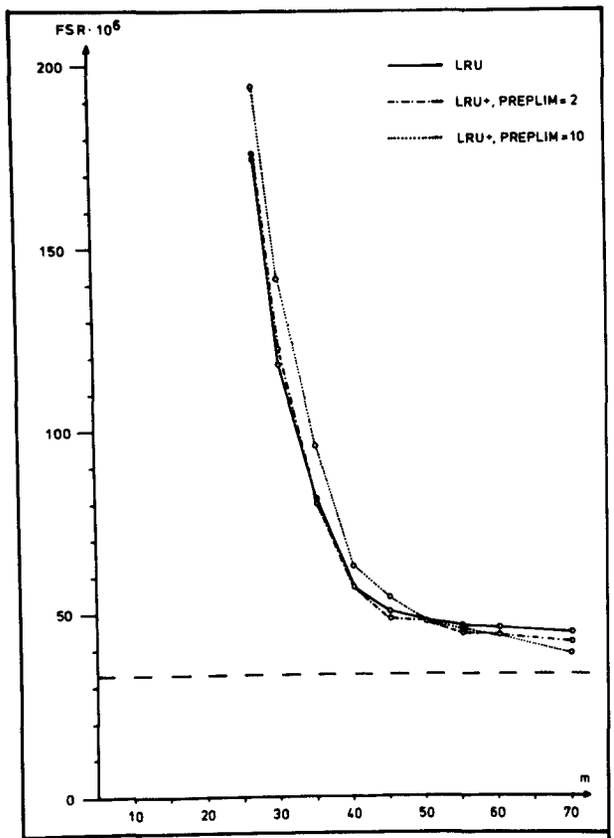


Bild 4.15
P5: Performance der Verfahren LRU und LRU+

Basis - LRU - Verfahren auf, bei starkem, d. h. risikoreichem Prepaging, kommt es aufgrund der Unzuverlässigkeit der Verkettungsinformation jedoch zu einer deutlichen Verschlechterung der Performance.

(b) DWS, DWS+

Die Ergebnisse beim DWS-Verfahren stimmen voll mit denen beim LRU-Verfahren überein, sowohl die Aussagen zu den Programmen P1 bis P4 betreffend als auch bezüglich der Sonderstellung von Programm P5. Auffällig ist die relativ geringe quantitative Verbesserung der Performance durch Prepaging bei Programm P3. Dies liegt daran, daß bei Programm P3 das DWS-Verfahren außerordentlich gut funktioniert und nahezu optimale Ergebnisse liefert (vgl. Bild 4.8) und deshalb das Verbesserungspotential sehr gering ist.

Bei den reinen Algorithmen in den Programmen P1 bis P3 ist die Speicherbelegung für eine bestimmte Fehlseitenrate deutlich geringer als beim LRU-Verfahren; bei Programm P4, das durch intensive Stack-Operationen gekennzeichnet ist, und bei Programm P5 sind solche prinzipiellen Unterschiede nicht

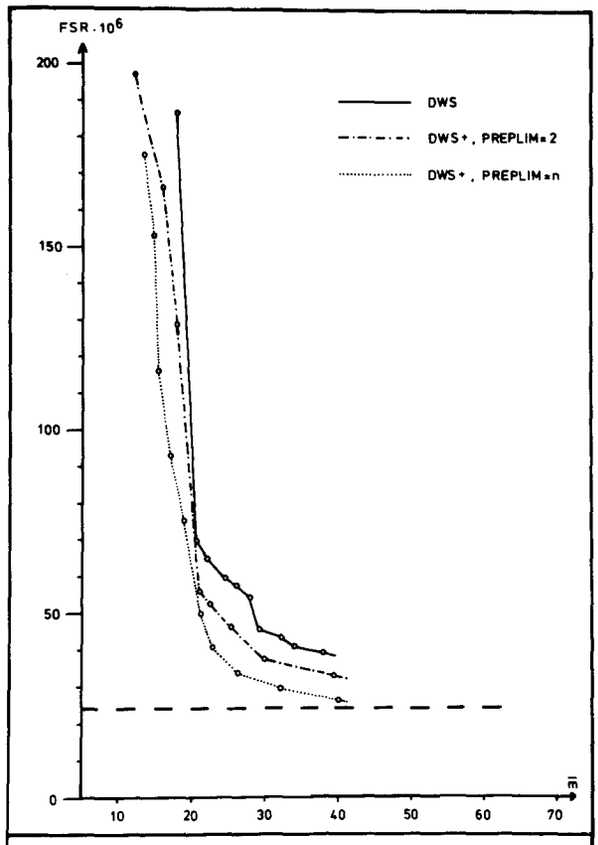


Bild 4.16
P1: Performance der Verfahren DWS und DWS+

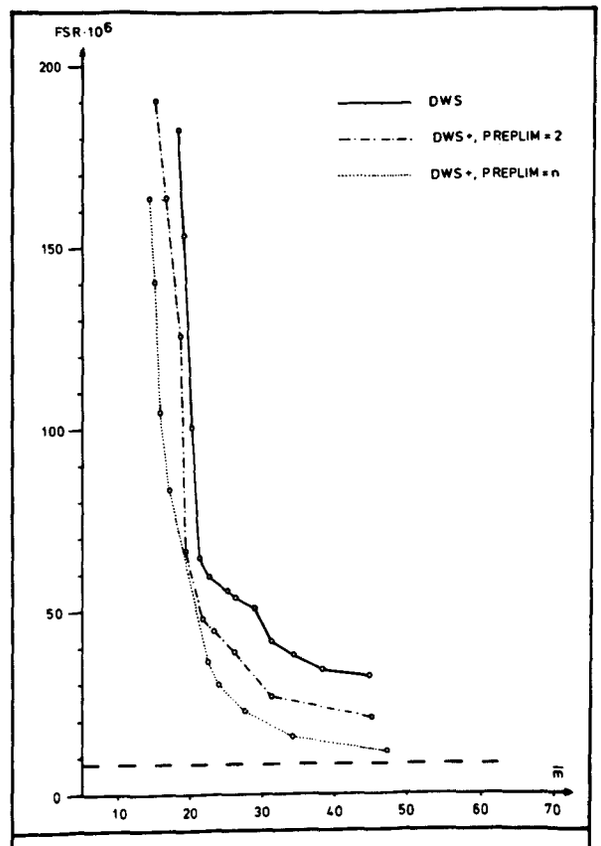


Bild 4.17
P2: Performance der Verfahren DWS und DWS+

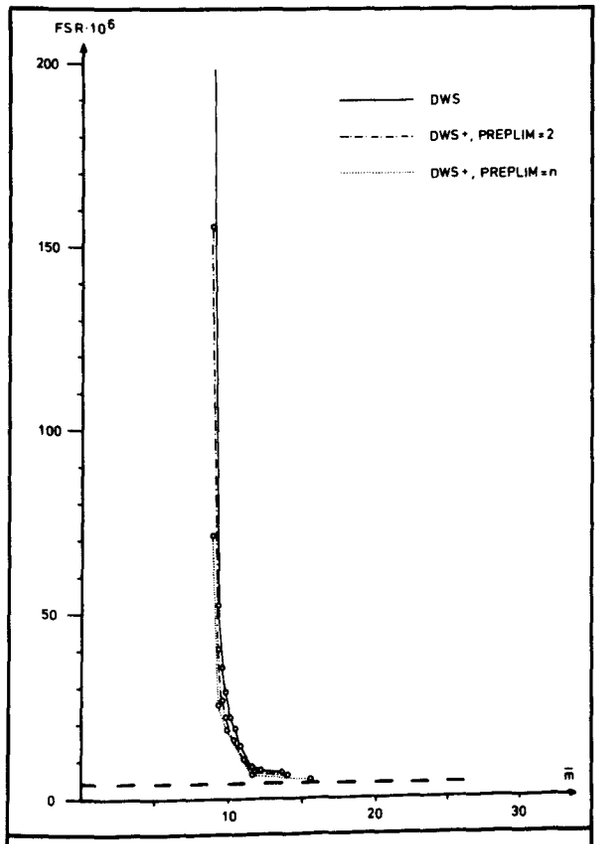


Bild 4.18
P3: Performance der Verfahren DWS und DWS+

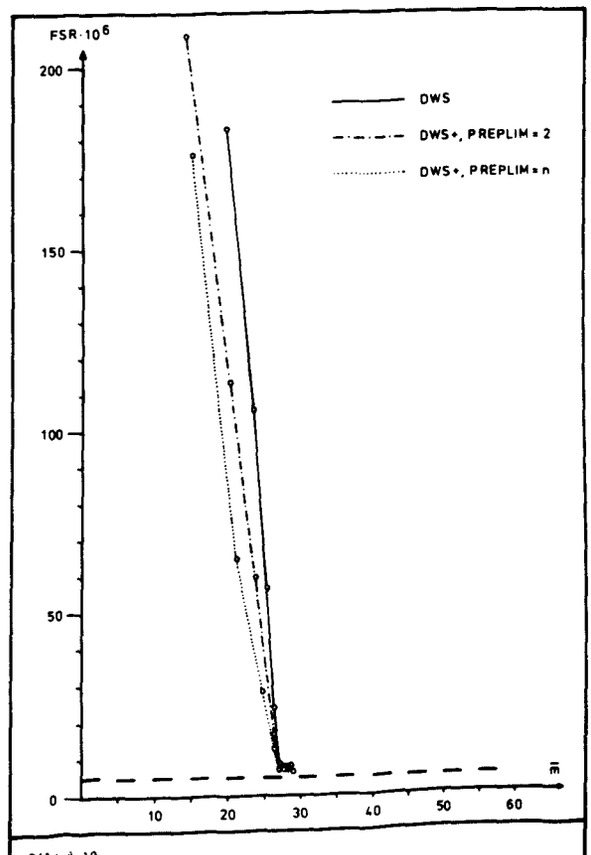
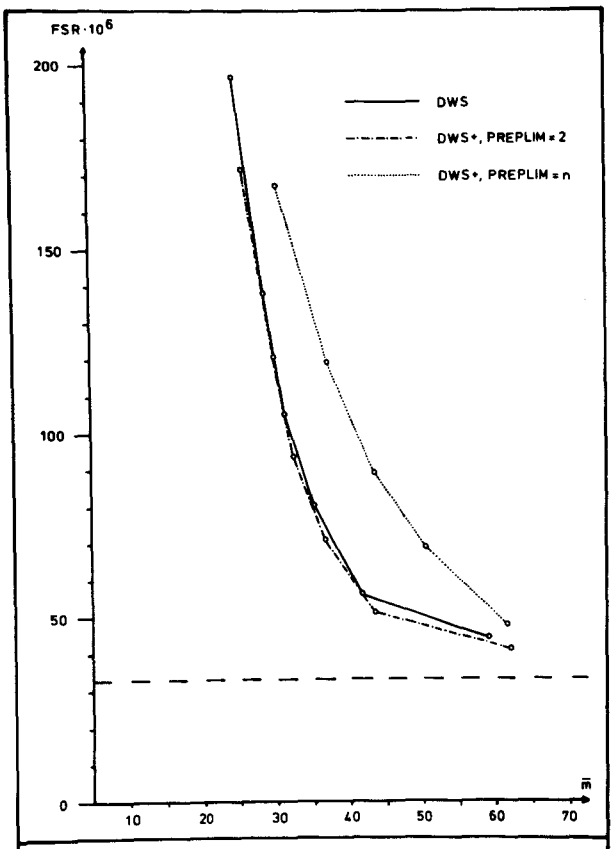


Bild 4.19
P4: Performance der Verfahren DWS und DWS+



B11d 4.20
P5: Performance der Verfahren DWS und DWS+

feststellbar. Interessant im Sinne des Themas ist, daß diese verfahrenstypischen Unterschiede bei den darauf aufbauenden Prepaging Verfahren erhalten bleiben.

(c) PFF, PFF+

Der Vergleich der Verfahren PFF und PFF+ liefert keine neuen Erkenntnisse: Die Ergebnisse decken sich qualitativ vollständig mit denen beim LRU- und DWS-Verfahren und stimmen mit denen des DWS-Verfahrens auch quantitativ gut überein; die Performance ist durchweg geringfügig schlechter als beim DWS-Verfahren. Der durch die Standard-Parameter abgedeckte Bereich der Fehlseitenrate schwankt beim PFF-Verfahren stärker als beim DWS-Verfahren.

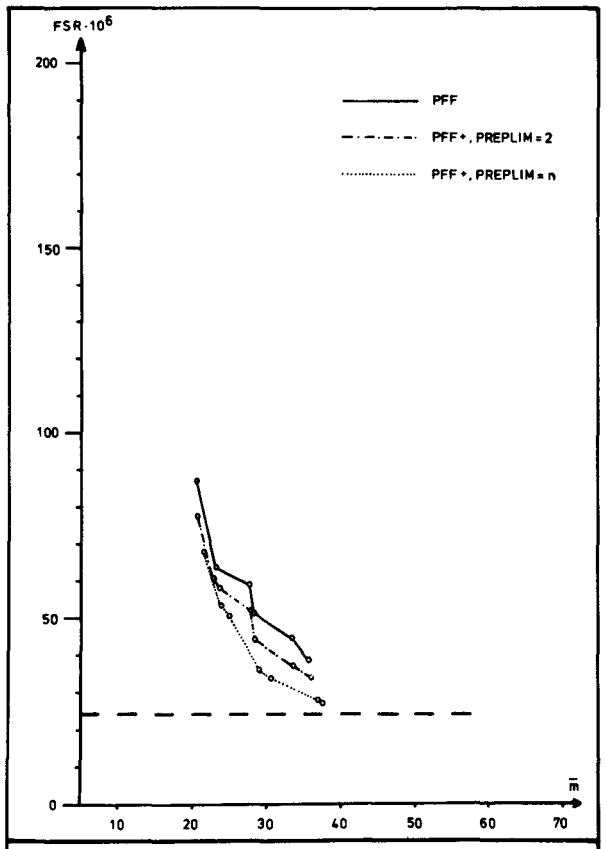


Bild 4.21
P1: Performance der Verfahren PFF und PFF+

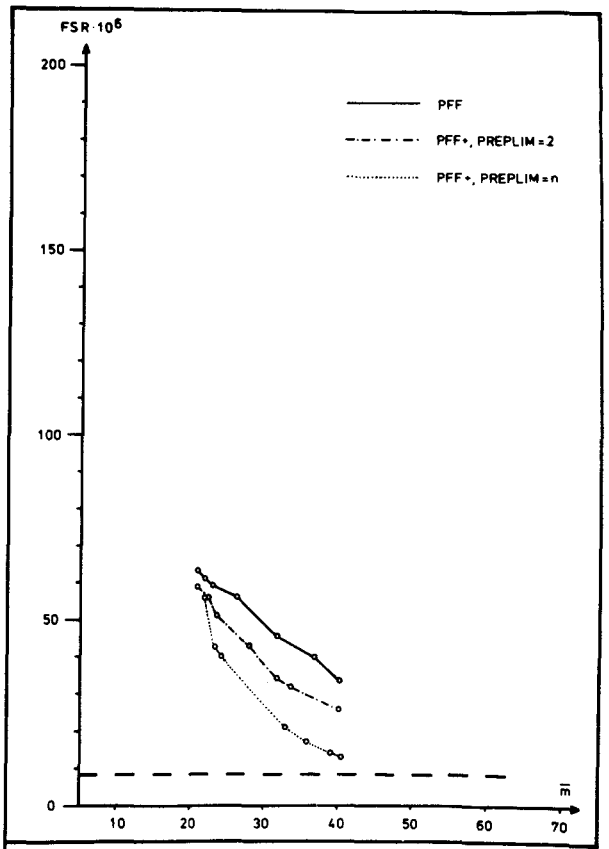


Bild 4.22
P2: Performance der Verfahren PFF und PFF+

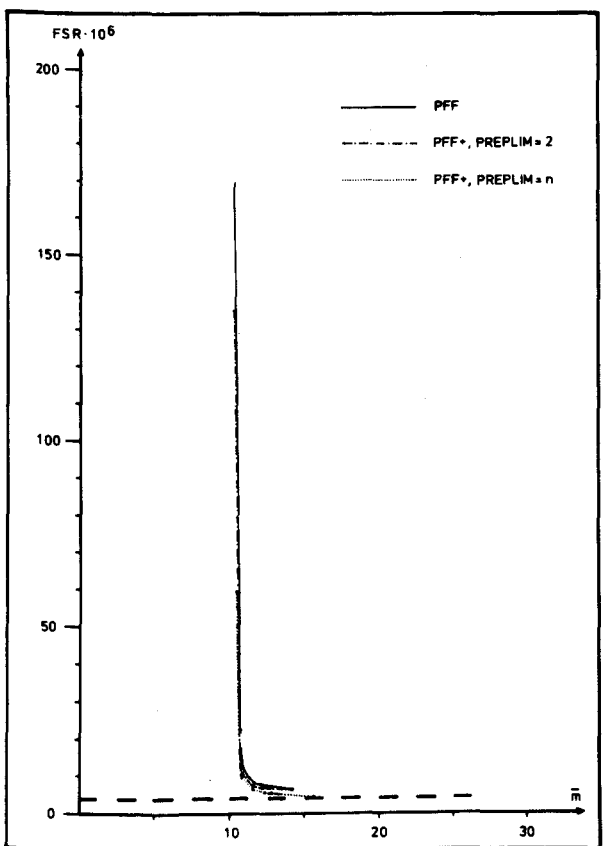


Bild 4.23
P3: Performance der Verfahren PFF und PFF+

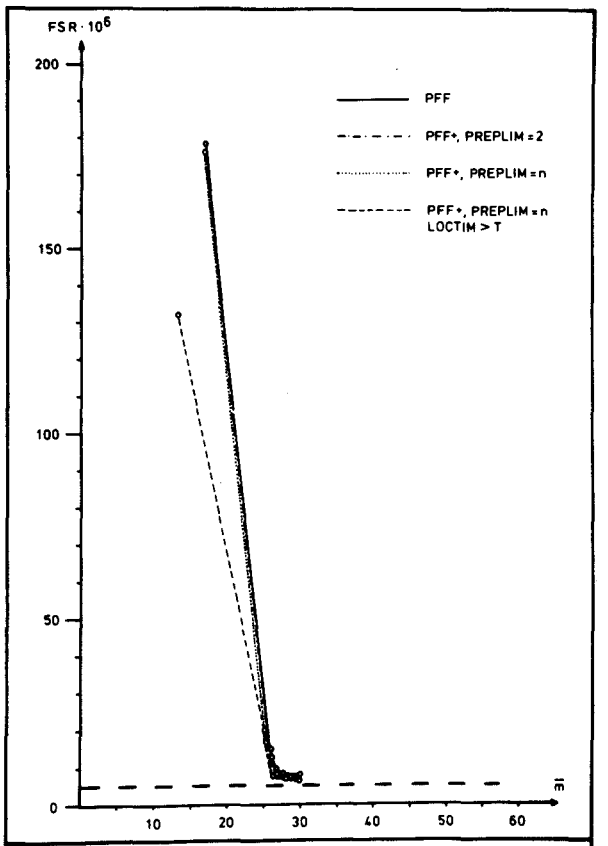


Bild 4.24
P4: Performance der Verfahren PFF und PFF+

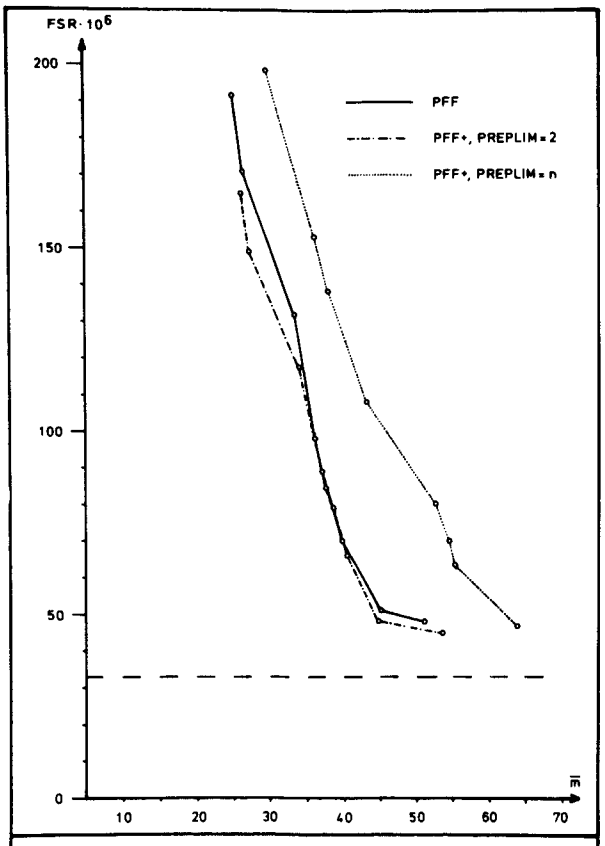


Bild 4.25
PS: Performance der Verfahren PFF und PFF+

Beim DWS- und beim PFF-Verfahren wird die Zeitsteuerung jeweils in der Logik des Basisverfahrens durchgeführt und es liegt nahe, auch quantitativ die Verkettungszeit in Übereinstimmung mit der jeweiligen Fenstergröße zu wählen. Im Einzelfall muß dies jedoch nicht zu optimalen Ergebnissen führen, wie ein Test beim PFF-Verfahren (Bild 4.24) zeigt, bei dem die Verkettungszeit größer als der Verfahrensparameter gewählt wurde. Systematische Untersuchungen zu dieser Fragestellung wurden nicht durchgeführt.

Ein wichtiges Ergebnis dieser Meßreihe ist, daß die Performance-Änderungen beim Übergang von Demand Paging zu Prepaging fast unabhängig vom Basisverfahren sind; qualitativ ergibt sich überhaupt kein Unterschied und auch quantitativ sind die Veränderungen ähnlich.

Diese Übereinstimmung in den Ergebnissen bei den verschiedenen Verfahren gestattet

es, die nachfolgende genauere Performance-Analyse und die Spezialuntersuchungen der folgenden Kapitel auf das LRU-Verfahren und die davon abgeleiteten Prepaging Verfahren zu beschränken. Dies ist nicht in erster Linie deshalb wünschenswert, weil dadurch der erforderliche Aufwand reduziert werden kann (tatsächlich wurden fast alle Messungen auch am DWS-Verfahren durchgeführt um sicherzustellen, daß sich keine prinzipiellen Unterschiede in den Aussagen ergeben), sondern weil nur bei Verfahren fester Speichergröße exakte Vergleichsmöglichkeiten gegeben sind. Ein Vergleich ist nur auf der Basis gleicher Speicherbelegung möglich; die Speicherbelegung ist aber bei Verfahren variabler Speichergröße selbst eine (vom Verfahrensparameter T) abhängige Größe, so daß zwischen den sich ergebenden mittleren Arbeitsspeichergrößen $\overline{m}$ die Werte der Performance-Maße interpoliert werden müssen, um Vergleiche durchführen zu können. Bei linearer Interpolation können an Stellen starker Steigungsänderung - und solche besitzen die Performance-Kurven in der Regel, da sie bei fallendem $\overline{m}$ bis zu einem kritischen Wert i. a. flach verlaufen und dann steil ansteigen - nicht nur quantitative, sondern sogar qualitative Veränderungen der relativen Performance entstehen. Gravierender als ein solcher, auf eine unzulängliche Interpolation zurückzuführender Fehler ist, daß möglicherweise zwischen zwei Intervallendpunkten die Performance-Kurve überhaupt nicht existiert, weil sie nur an diskreten Punkten definiert ist. Die mittlere Speicherbelegung $\overline{m}$, die selbst wieder abhängige Veränderliche des Verfahrensparameters T ist, der seiner Natur nach diskret ist, ist als Funktion von T stückweise konstant und nimmt nur relativ wenige verschiedene Werte an; nur für solche Werte ist $FSR(\overline{m})$ definiert.

Aus diesen Gründen werden vergleichende Messungen am LRU-Verfahren vorgenommen, bei dem solche Probleme nicht existieren. Für die Sonderuntersuchungen werden vorwiegend die Programme P1, P4 und P5 verwendet, P1 als Normalprogramm, P4 als besonders Prepaging - geeignetes exotisches Programm und P5 als ein schlecht geeignetes Programmbeispiel.

Um die Performance des Basisverfahrens mit der Performance der Prepaging Verfahren vergleichen zu können, wurden die Daten des Demand Verfahrens (für FSR und MFSR) mit 100 % bewertet und die Werte der Prepaging Verfahren darauf bezogen. Die Ergebnisse sind für die Programme P1, P2, P4 und P5 in den Bildern 4.26 bis 4.29 dargestellt.

Es ist zu beachten, daß bei dieser Darstellung bei kleinen Fehlseitenraten (also großen Speichern) notwendigerweise die Performance-Unterschiede gering sind, weil bei beiden zu vergleichenden Verfahren die Anzahl der Fehlseitenbedingungen dominiert wird durch die als additive Konstante zu betrachtende Anzahl der primären Faults, so daß bei Quotientenbildung das Ergebnis nur geringfügig von 1 abweichen kann. Dies wird den subjektiven Qualitäten der Verfahren nicht ganz gerecht, wie im nachfolgenden Beispiel erläutert wird: Ein Programm der Größe n = 200 Seiten liefere bei einem Basisverfahren 222 Page Faults; wenn das gleiche Programm bei einem Prepaging Verfahren 200 Page Faults liefert, so ist das eine Verbesserung der relativen Performance um bescheidene 10 % gegenüber dem Basisverfahren; gleichzeitig ist diese Verbesserung jedoch die maximal erreichbare und das Ergebnis somit optimal. Dieser Effekt läßt sich vermeiden, wenn man unter Weglassung der primären Faults die prinzipiell vermeidbaren Faults in Beziehung setzt, was jedoch andere Nachteile mit sich bringt (vgl. Kap. 4.3.4), unter anderem den, daß dann die objektive Bedeutung der Ergebnisse verzerrt wird, weil z. B. der Wert (oder Schaden) eines Page Faults davon abhängig ist, ob er vermeidbar oder unvermeidbar ist oder auch, ob es einer von 10 oder von 100 ist. Die vorher geschilderten Eigenschaften der gewählten Darstellung sollten bei der Wertung der Ergebnisse berücksichtigt werden.

Bei allen Programmen zeigen die Kurven bei den verschiedenen Performance-Maßen FSR und MFSR sehr große Ähnlichkeiten, das Niveau ist jedoch verschieden. Außer bei P5 ist die Performance der Prepaging Verfahren bzgl. der Fehlseitenrate immer besser als beim Demand Verfahren und das Bild ändert sich bis auf eine kleine Ausnahme bei P1 auch nicht bezüglich der modifizierten Fehlseitenrate. Auffällig bei diesen

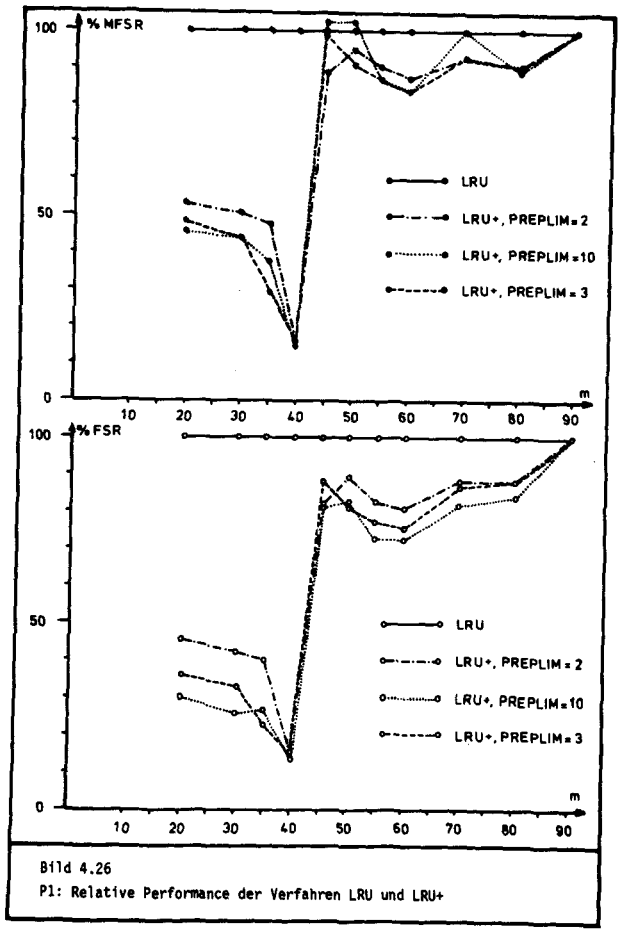


Bild 4.26
P1: Relative Performance der Verfahren LRU und LRU+

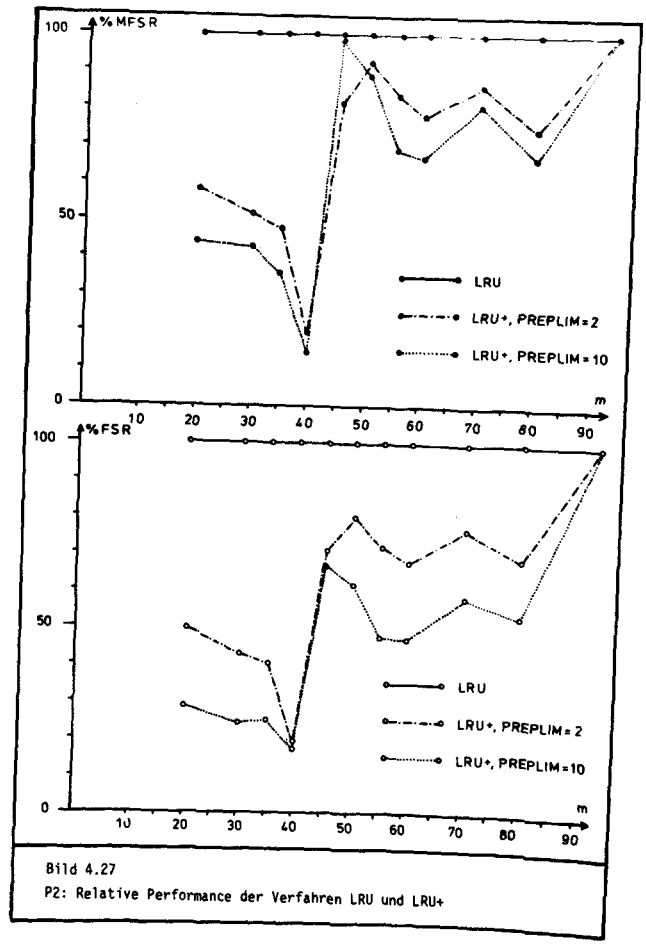


Bild 4.27
P2: Relative Performance der Verfahren LRU und LRU+

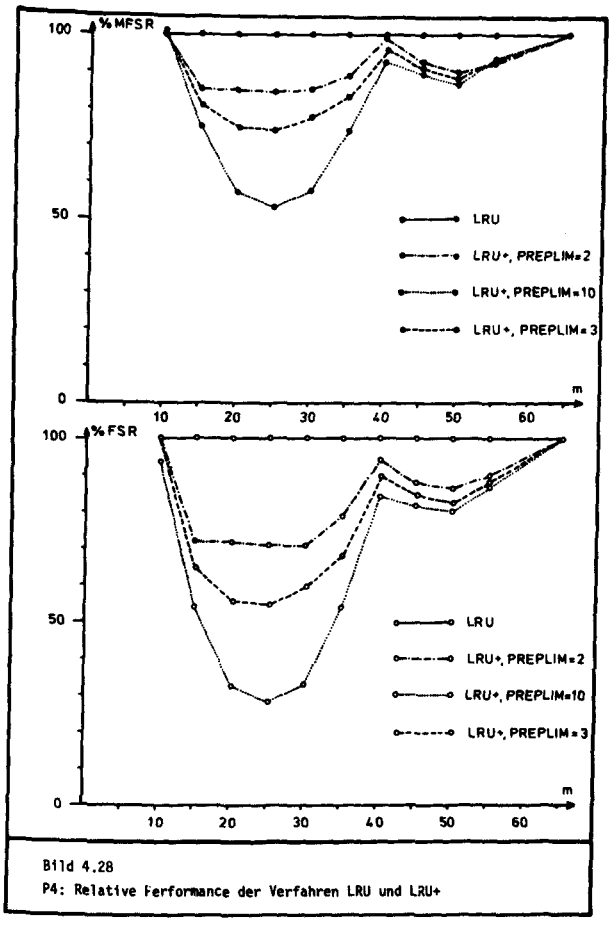


Bild 4.28
P4: Relative Performance der Verfahren LRU und LRU+

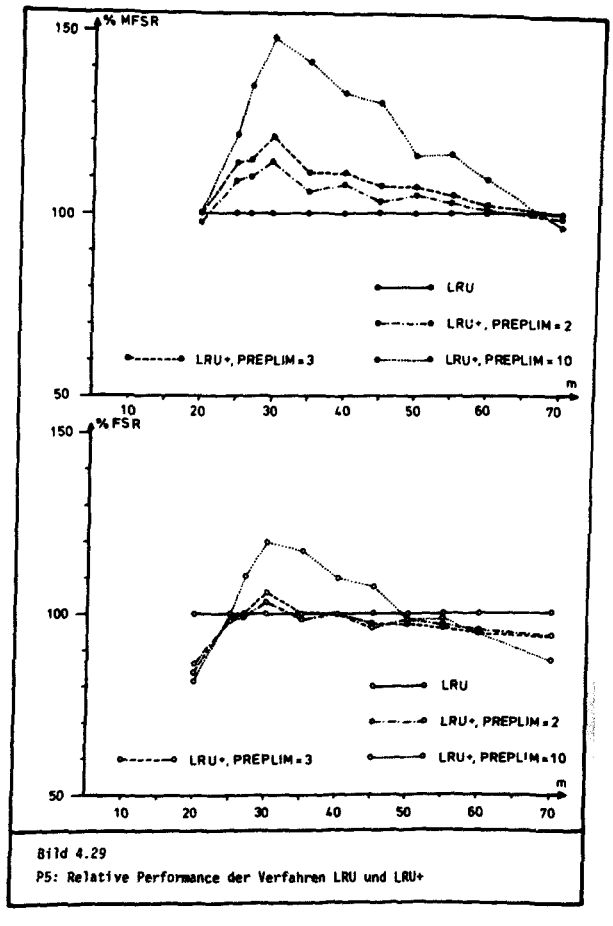


Bild 4.29
P5: Relative Performance der Verfahren LRU und LRU+

Programmen sind die Einbruchstellen, das sind die Bereiche, in denen die Fehlseiten der Prepaging Verfahren nur einen kleinen Bruchteil derjenigen des Basisverfahrens ausmachen. Nimmt man die absoluten Darstellungen der Bilder 4.11 bis 4.14 zu Hilfe, so erkennt man, daß diese günstigen Ergebnisse in den Bereich des steilen Anstiegs der Fehlseitenrate fallen. d. h., bei den Prepaging Verfahren beginnt dieser Anstieg erst bei kleineren Speichergrößen und ist auch nicht so ausgeprägt wie bei den Demand Verfahren. Dies bedeutet, daß das Verhalten bezüglich Trashing bei den Prepaging Verfahren unkritischer ist als bei den Demand Verfahren.

Ein völlig anderes Bild liefert das Programm P5. Die Fehlseitenraten für PREPLIM = 2 und PREPLIM = 3 verlaufen noch verhältnismäßig günstig; die modifizierte Fehlseitenrate zeigt jedoch, daß diese recht bescheidenen Ergebnisse durch relativ viele Page In's erkauft wurden. Völlig unbefriedigend sind die Ergebnisse für PREPLIM = 10. Die vielen durch Prepaging geladenen Seiten (große Werte für MFSR) sind, wie die ebenfalls schlechten Werte für FSR zeigen, offensichtlich nur zufällig tatsächlich benötigte Seiten. Die Ergebnisse belegen sehr nachdrücklich, daß Prepaging auch bei ausschließlicher Berücksichtigung der Page Faults keinesweges zu einer Verbesserung der Performance führen muß.

Eine weitere Untersuchung betrifft das Prepaging Risiko. Dazu sind in den Bildern 4.30 bis 4.33 für die gleiche Messung die relativen Ergebnisse jeweils für FSR und MFSR gegenübergestellt. Die Differenz zwischen den beiden Performance-Maßen - das ist der schraffierte Bereich in den Bildern - ist ein Maß für das Prepaging-Risiko.
Augenfällig ist, daß das Prepaging Risiko für PREPLIM = 2 deutlich kleiner ist als für PREPLIM = 10, wie es nicht anders zu erwarten war. Bei den Programmen P1 bis P4 resultiert aus diesem erhöhten Risiko eine deutlich verringerte Fehlseitenrate und in den meisten Fällen verläuft auch für das Performance-Maß MFSR die Kurve für PREPLIM = 10 günstiger als für PREPLIM = 2 (vgl. Bilder 4.26 bis 4.29), was bedeutet, daß das Risiko durch zusätzlich Page In's weniger stark ansteigt als der Gewinn durch Reduktion der Page Faults.

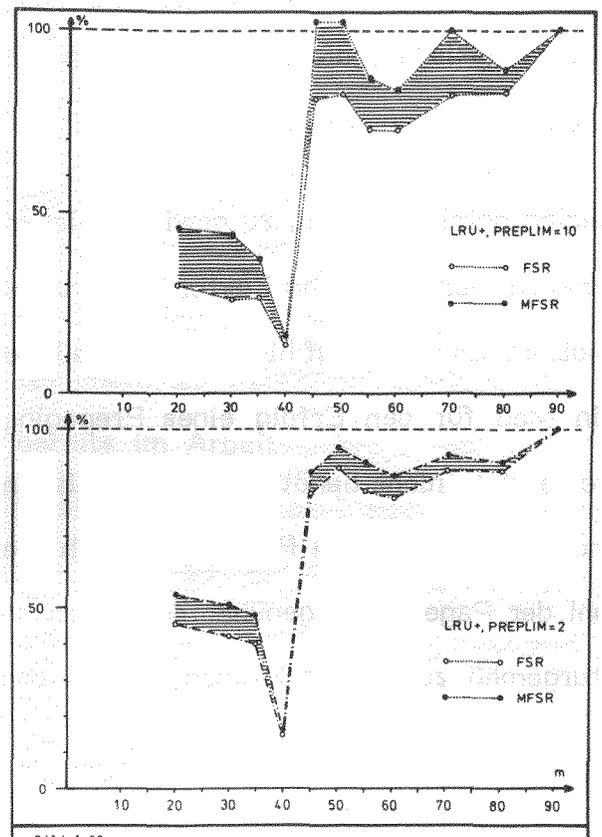


Bild 4.30
P1: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM=2/10

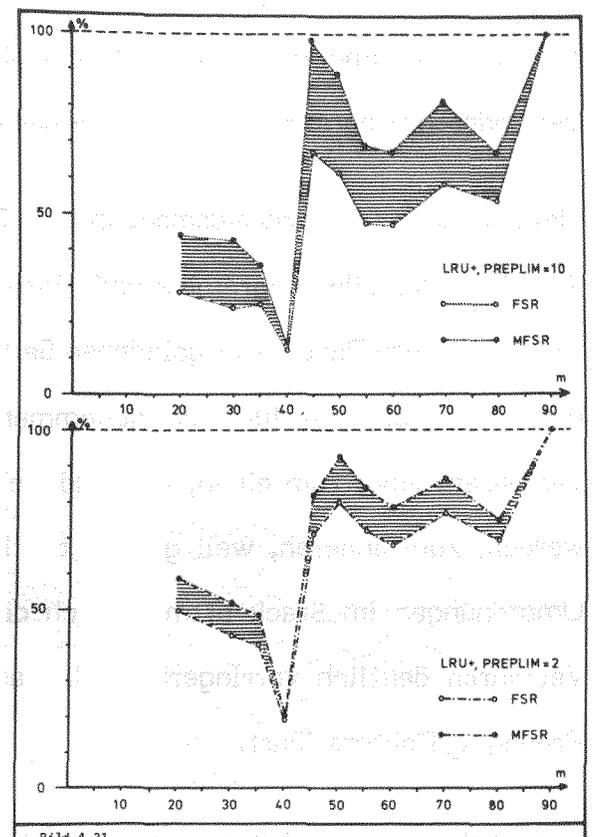


Bild 4.31
P2: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM=2/10

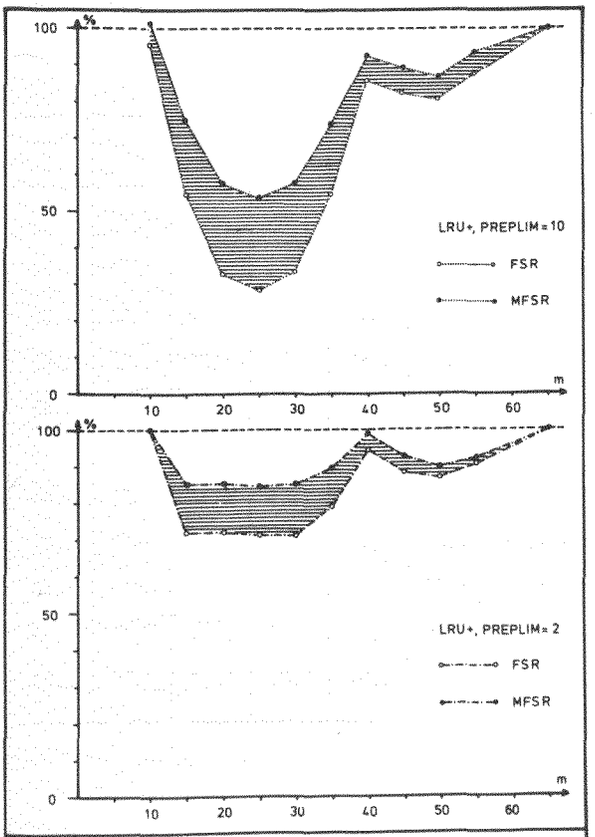


Bild 4.32
P4: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM=2/10

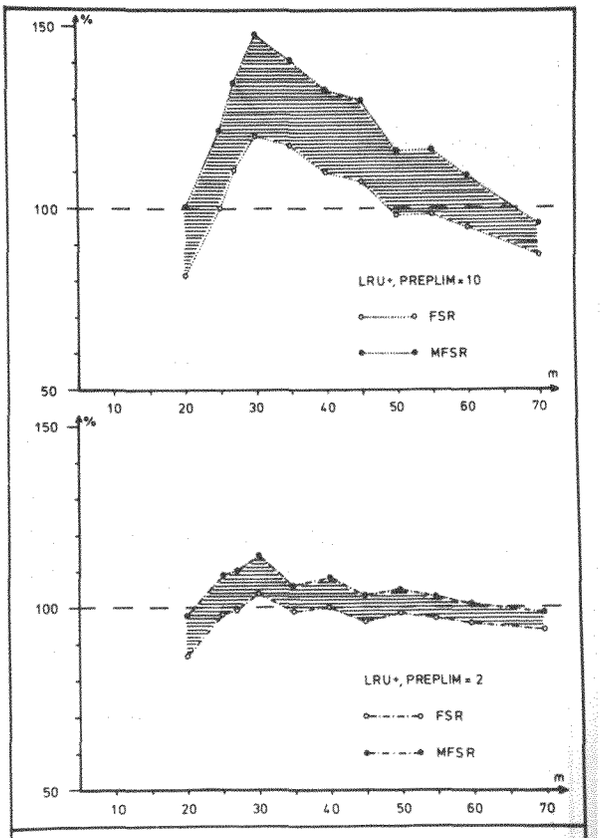


Bild 4.33
P5: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM=2/10

Für das Programm P5 zeigen sich sehr gleichförmige Risiken, die eher über denjenigen der anderen Programme liegen, ohne daß dadurch ein Gewinn erzielt wird.

Um Aussagen über das Ausmaß, in dem Prepaging getrieben wird, zu gewinnen, ist in Tab. 4.14 für alle Programme und Parameterwerte der Prepaging Faktor (das ist die Anzahl der pro Page Fault geladenen Seiten) notiert. Es ist darauf hinzuweisen, daß die Prepaging Faktoren für sich genommen kein Maß für den Erfolg eines Prepaging Verfahrens sind, zum einen, weil möglicherweise viele nicht benötigte Seiten geladen werden, zum anderen, weil gerade bei besonders erfolgreichem Prepaging durch die Umordnungen im Stack oftmals auch die Zahl der Page In's gegenüber dem Demand Verfahren deutlich verringert wird, was naturgemäß zu relativ kleinen Werten des Prepaging Faktors führt.

Zunächst fällt auf, daß der Prepaging Faktor durchweg zwischen 1 und 2 liegt, immer kleiner als 4 ist und Werte zwischen 3 und 4 bereits Ausnahmen sind.

PREPLIM / Index der Speicher-größen-Vorgabe	P1			P2		P3		P4			P5		
	2	3	10	2	10	2	10	2	3	10	2	3	10
1	1,38	1,76	2,32	1,40	2,39	1,48	1,74	1,01	1,01	1,02	1,27	1,40	1,52
2	1,46	1,76	2,92	1,50	3,18	1,54	2,59	1,41	1,57	1,91	1,24	1,34	1,47
3	1,44	1,67	1,97	1,44	2,08	1,41	2,74	1,39	1,79	3,15	1,22	1,33	1,50
4	1,15	1,21	1,37	1,12	1,46	1,27	1,48	1,42	1,82	3,58	1,22	1,30	1,53
5	1,17	1,27	1,60	1,32	2,14	1,42	2,23	1,41	1,67	3,04	1,15	1,22	1,45
6	1,13	1,25	1,53	1,36	2,11	1,13	1,22	1,28	1,48	1,88	1,17	1,23	1,45
7	1,19	1,28	1,43	1,36	2,15	1,18	1,49	1,09	1,15	1,21	1,16	1,21	1,46
8	1,12	1,24	1,34	1,33	2,08	1,09	1,18	1,09	1,15	1,19	1,14	1,22	1,41
9	1,09	1,12	1,21	1,27	1,93	1,00	1,00	1,07	1,12	1,15	1,13	1,19	1,40
10	1,05	1,07	1,11	1,18	1,57	1,00	1,00	1,05	1,10	1,13	1,11	1,18	1,33
11	1,00	1,00	1,00	1,00	1,00	-	-	1,01	1,01	1,01	1,10	1,15	1,23

Tab. 4.14

Prepaging Faktoren beim LRU+ - Verfahren

Es ist zunächst nicht einsichtig, daß die Prepaging Faktoren mit fallenden Speichergrößen wachsen, da mit der Arbeitsspeichergröße auch die Verkettungszeiten verkleinert werden, wodurch sich zwangsläufig die mittleren Verkettungslängen verringern, was sich zumindest bei unlimitiertem Prepaging in einer Verkleinerung des Prepaging Faktors niederschlagen müßte. Die Erklärung dafür ist, daß bei größeren Arbeitsspeichern die Wahrscheinlichkeit, daß sich Seiten von aktuell gewordenen Lokalitäten bereits im Arbeitsspeicher befinden, stark anwächst, so daß immer nur ein relativ kleiner Teil der neuen Lokalitäten vom Sekundärspeicher geladen werden muß.

Die Werte für den Prepaging Faktor sind über die Programmlaufzeit gemittelt; die programmtypischen Prepaging Faktoren liegen höher, da am Anfang keine Verkettungsinformationen vorliegen und erst allmählich - so wie diese aufgebaut werden - Prepaging betrieben werden kann und die Prepaging Faktoren von 1 auf ihren programmtypischen Wert anwachsen. Da P2 eine Fortsetzung von P1 ist (vgl. Kap. 4.3.2), können die programmtypischen Endwerte der Prepaging Faktoren berechnet werden; sie liegen maximal 20 - 25 % über den mittleren Werten, ändern das Bild also nicht nennenswert.

Die meist unter 2 liegenden Werte der Prepaging Faktoren und die geringen Zuwächse durch Erhöhung der PREPLIM-Werte von 2 bzw. 3 auf 10 zeigen, daß nur sehr selten eine größere Anzahl von Seiten pro Page Fault geladen wird (was die bereits vorher getroffene Feststellung, daß PREPLIM = 10 praktisch keine Beschränkung des Prepaging bedeutet, rechtfertigt). Da nun gerade weitgehendes Prepaging aufgrund langer Verkettungen das Risiko erhöht, scheint eine Begrenzung des Prepaging zum Abbau der Extreme (Risiken, aber auch Chancen) grundsätzlich angebracht. Die Bilder 4.26, 4.28 und 4.29, in die die Performance-Kurven für PREPLIM = 3 eingetragen sind, zeigen, daß für PREPLIM = 3 die Vorteile der Prepaging Verfahren angemessen zum Tragen kommen, die Risiken aber beschränkt bleiben, was vor allem an Programm P5 sichtbar wird, aber auch aus dem vergleichsweise guten Abschneiden beim Performance-Maß MFSR hervorgeht. Bei Programm P4, bei dem der Prepaging Faktor mehrfach über 3 liegt, wäre ein etwas größeres Limit besser.

Zusammenfassend kann festgestellt werden, daß eine grundsätzliche Begrenzung des Prepaging über die natürliche Begrenzung durch die Verkettungslängen hinaus sinnvoll ist und daß PREPLIM = 3 ein guter Kompromiß ist, wenn eine einheitliche Vorgabe für alle Programme gemacht werden soll; denkbar und erfolgversprechender wäre eine individuelle Festlegung dieses Parameters für jedes Programm, wodurch insbesondere die Risiken bei nicht rekursiven Programmen gering gehalten werden könnten, ohne potentielle Vorteile bei gut geeigneten Programmen zu verschenken.

4.3.2 Einfluß der Programmlänge

Es wurde bereits in Kap. 3 darauf hingewiesen, daß aus Aufwandsgründen immer nur kurze Zeitintervalle echter Programmlaufzeit simuliert werden können. Um dennoch Aussagen über Tendenzen im Performance-Verhalten längerer Programme zu gewinnen, wurde Programm P2 als Verlängerung des Programmes P1 ausgelegt. Hierbei wurden nicht unterschiedlich lange Abschnitte der gleichen Referenzkette verwendet, sondern eine logische Verlängerung von Programm P1 vorgenommen, indem durch eine kleine Ablaufsteuerung das aus der IMSL stammende Programm zur Matrix-Inversion dreimal aufgerufen wurde, um die zuvor durch Zufallszahlen erzeugten Matrizen zu verarbeiten; von diesem Programmlauf wurde eine eigene Referenzkette erzeugt. Diese Vorgehensweise entspricht in kleinem Maßstab dem, was typischerweise bei rechenintensiven Programmen geschieht, nämlich, daß über eine Ablaufsteuerung funktionsorientierte Programmteile mit veränderten Eingabedaten immer wieder ablaufen.

Die Bilder 4.11 / 4.12, 4.16 / 4.17 und 4.21 / 4.22 in Kap. 4.3.1 zeigen für die Verfahrensgruppen LRU, DWS und PFF übereinstimmend einen gleichartigen Verlauf der Performance-Kurven, wobei bei P2 der Kurvenverlauf meist etwas glatter ist und vor allem die Abstände der Performance-Kurven gegenüber P1 vergrößert sind, die Performance-Unterschiede zwischen den einzelnen Verfahren also größer geworden sind.

Die schraffierten Bereiche in den Bildern 4.30 und 4.31 in Kap. 4.3.1 zeigen, daß das Prepaging Risiko bei Programm P2 nicht - wie man bei oberflächlicher Betrachtung erwarten könnte - geringer geworden ist, sondern im Gegenteil vergrößert ist. Bei genauerer Analyse ist dies jedoch aus zwei Gründen verständlich:

(a) Es gibt keine Anhaltspunkte dafür, daß eine zu einem bestimmten Zeitpunkt erstellte Verkettung bei längeren Programmen sicherer ist als bei kurzen; mit der Programmlaufzeit steigt im Gegenteil die Wahrscheinlichkeit, daß eine Verkettung revidiert werden muß; dadurch dürfte der Vorteil aufgewogen werden, der bei längeren Programmen wegen der auf die Programmlänge bezogenen Angaben dadurch entsteht, daß sich Veränderungen relativ gesehen weniger stark auswirken.

(b) Da alle Messungen bei informationslosem Anfangszustand des Verkettungsvektors begonnen wurden, müssen zunächst Verkettungen aufgebaut werden, bevor diese in Form von Prepaging genutzt werden können. Dies bedeutet, daß erst allmählich mit zunehmendem Programmfortschritt überhaupt Prepaging betrieben werden kann und auch erst dann ein Risiko entsteht. Da diese risikofreie bzw. mit geringem Risiko behaftete Anfangsphase bei Programm P1 relativ zur Programmlaufzeit länger dauert als bei Programm P2, muß das über die Laufzeit gemittelte Risiko geringer sein als bei Programm P2.

Die Kehrseite des unter (b) vorgebrachten Argumentes ist, daß, weil während der relativ länger dauernden Anfangsphase kaum Prepaging betrieben wird, auch der mögliche Nutzen kleiner sein muß. Dies wird durch die Bilder 4.34 und 4.35 voll bestätigt. In diesen Bildern ist für PREPLIM = 2 und PREPLIM = 10 relativ zum jeweiligen Basisverfahren die Performance der Prepaging Verfahren für P1 und P2 gegenübergestellt, und zwar für die Performance-Maße FSR und MFSR.
Mit einer kleinen Ausnahme für PREPLIM = 2 und extrem kleine m (d.h. weit im

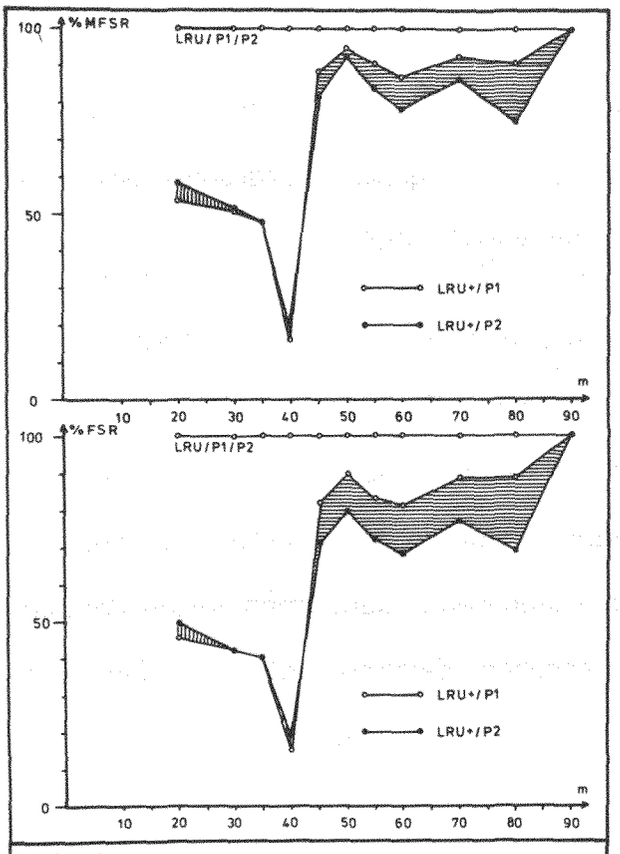


Bild 4.34
Vergleich der relativen Performance der Programme P1 und P2 für das Verfahren LRU+, PREPLIM=2

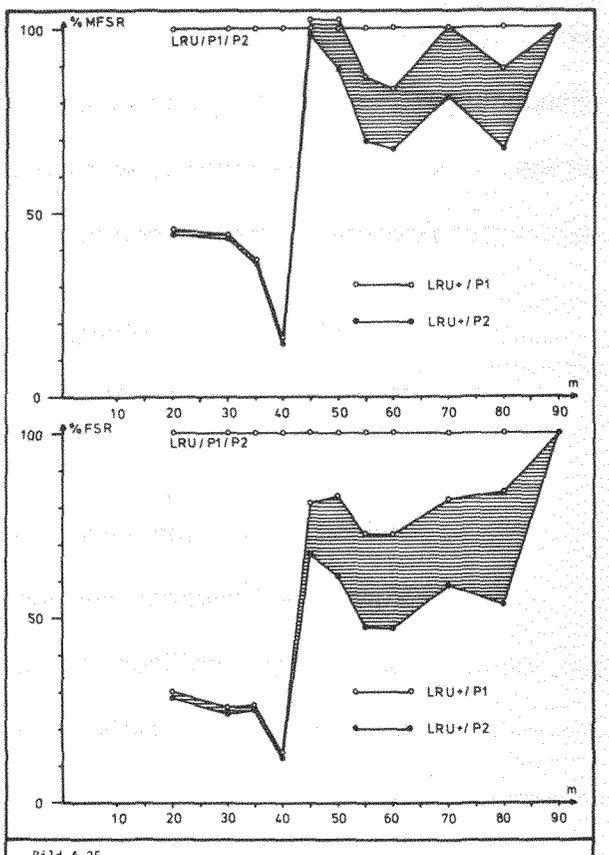


Bild 4.35
Vergleich der relativen Performance der Programme P1 und P2 für das Verfahren LRU+, PREPLIM=10

Thrashing-Bereich) ist die relative Performance bei P2 gegenüber dem Basisverfahren stets deutlich besser als bei P1, dies gilt insbesondere auch für das Performance-Maß MFSR, was bedeutet, daß der Nutzen durch Prepaging bei Programm P2 gegenüber P1 noch stärker wächst als das Risiko.

Zusammenfassend läßt sich aufgrund der hier präsentierten Ergebnisse sagen, daß zu erwarten ist, daß die durch Prepaging erzielbaren Performance-Gewinne bei langlaufenden Programmen eher größer sein werden als bei den bei der Simulation benutzten relativ kurzen Referenzketten.

4.3.3 Speicherung der Verkettungsinformation

Bei allen bisherigen Messungen wurden die Simulationen mit informationslosem Verkettungsvektor begonnen. Dies ist unvermeidlich bei Programmen, die erstmalig zur Ausführung kommen. Am Ende eines Programmlaufes existieren jedoch in Form von Verkettungen Informationen über das Programmverhalten, und es liegt der Gedanke nahe, daß diese Informationen nicht einfach vernachlässigt, sondern gespeichert und bei einem erneuten Programmaufruf nutzbar gemacht werden sollten. Die praktischen Probleme, die die Aufbewahrung und sinnvolle Nutzung dieser Informationen in einem erneuten Programmlauf verursachen können, sollen hier nicht untersucht werden, sondern das Performance - wirksame Potential, das in dieser Maßnahme steckt. Dazu wurde der Verkettungsvektor vor einer erneuten Simulation mit den Werten initialisiert, die er am Ende einer vorher durchgeführten Simulation hatte.

Die Initialisierung des Verkettungsvektors ermöglicht vom Programmstart an Prepaging, wodurch sich insbesondere (bei großen Speicherplatzvorgaben) die Möglichkeit ergibt, daß die Anzahl der Fehlseitenbedingungen die Anzahl der Programmseiten unterschreitet, was bei Demand Paging oder Prepaging mit nicht initialisiertem Verkettungsvektor prinzipiell nicht möglich ist. Das höhere Maß an Prepaging führt jedoch nicht automatisch zu einer Verbesserung der Performance des Verfahrens: Da der Verkettungsalgorithmus die Verkettungsinformationen permanent überprüft und aktualisiert, könnte es sein, daß der Verkettungszustand am Programmende nicht sehr viel brauchbare Informationen für den Programmanfang enthält, so daß sich bei Nutzung dieser Informationen durch Prepaging die Performance verschlechtern würde. Wegen dieser Befürchtung wurde als Alternative auch der Zustand des Verkettungsvektors nach Abschluß der ersten Verkettung (Lokalität) gespeichert; es zeigte sich jedoch, daß zu diesem frühen Zeitpunkt der Verkettungsgrad so gering war, daß die Nutzung dieser Informationen schon von daher nur geringe Auwirkungen haben konnte. Diese Alternative wurde deshalb nicht weiter verfolgt, zumal die Messungen ergaben,

daß die Zweifel an der Brauchbarkeit der übernommenen Verkettungsinformationen nicht angebracht waren.

Die Messungen wurden für die Programme P1 und P5 mit dem LRU+ - und dem DWS+ - Verfahren durchgeführt, mit PREPLIM = 2 bei beiden Programmen und zusätzlich mit PREPLIM = 10 bei P1; die Bilder 4.36 und 4.37 zeigen die Ergebnisse, wobei zum Vergleich auch die Fehlseitenrate der Normalverfahren (Prepaging ohne Initialisierung des Verkettungsvektors) und der Basisverfahren eingezeichnet ist.

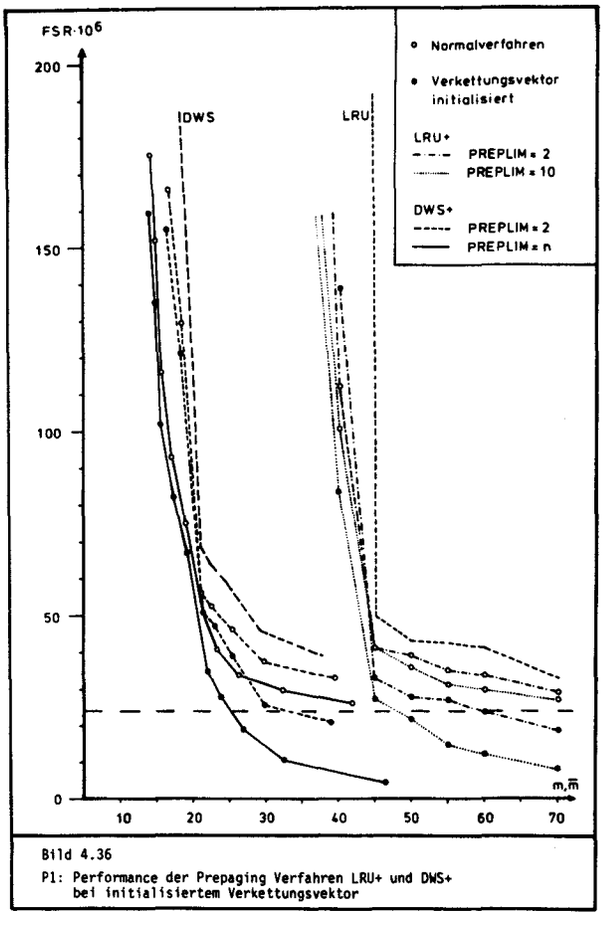


Bild 4.36
P1: Performance der Prepaging Verfahren LRU+ und DWS+ bei initialisiertem Verkettungsvektor

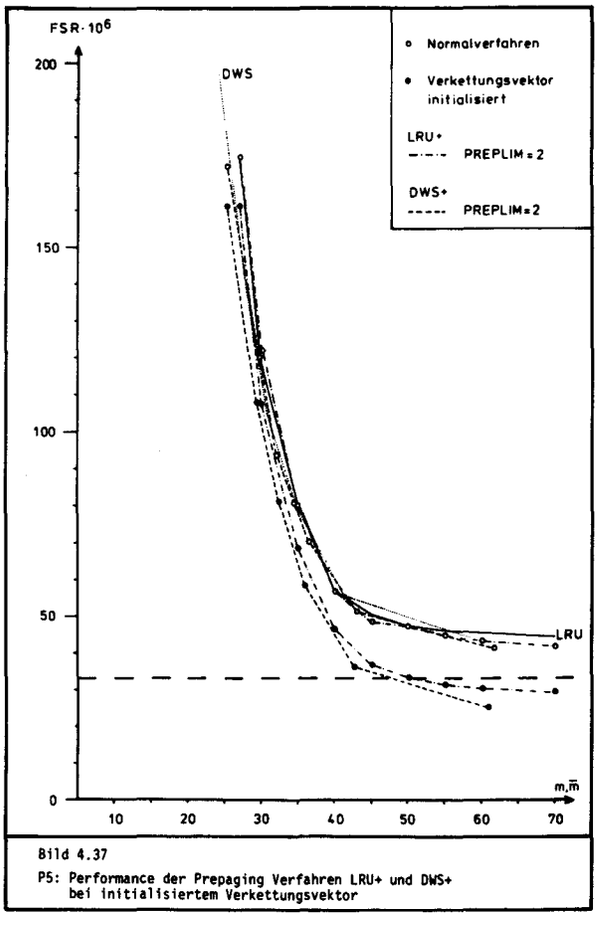


Bild 4.37
P5: Performance der Prepaging Verfahren LRU+ und DWS+ bei initialisiertem Verkettungsvektor

Die Bilder zeigen eine deutliche Performance-Verbesserung gegenüber den Normalverfahren, wobei dieses Mal das Programm P5 keine Ausnahme macht. Die Darstellungen der relativen Performance für das LRU-Verfahren in den Bildern 4.38 und 4.39 zeigen, daß die Verbesserungen gerade im für die Praxis relevanten Bereich kleiner Fehlseitenraten besonders groß sind, wobei die Verbesserungen beim Performance-Maß MFSR ebenso deutlich sind, was unterstreicht, daß sie nicht auf Kosten erhöhter Page-

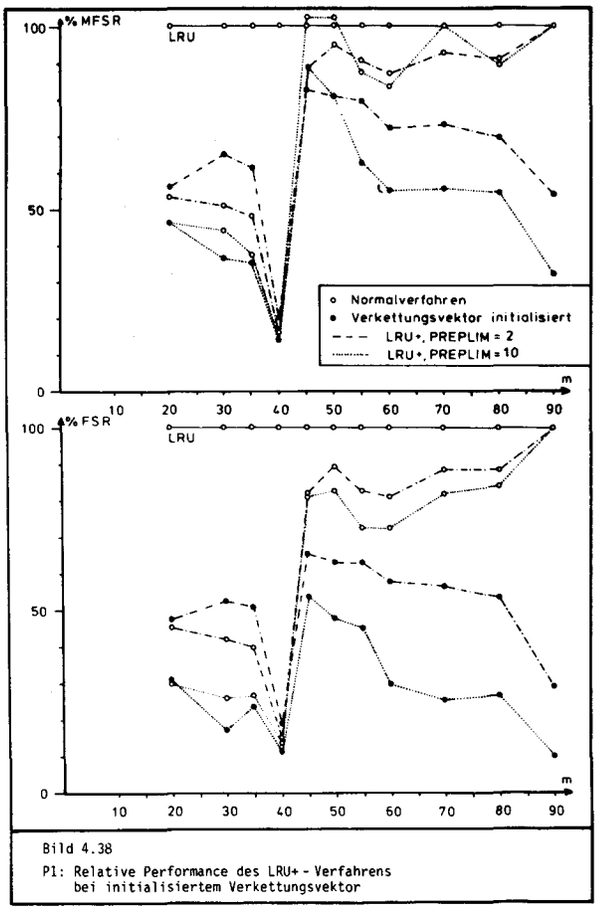


Bild 4.38
P1: Relative Performance des LRU+ - Verfahrens bei initialisiertem Verkettungsvektor

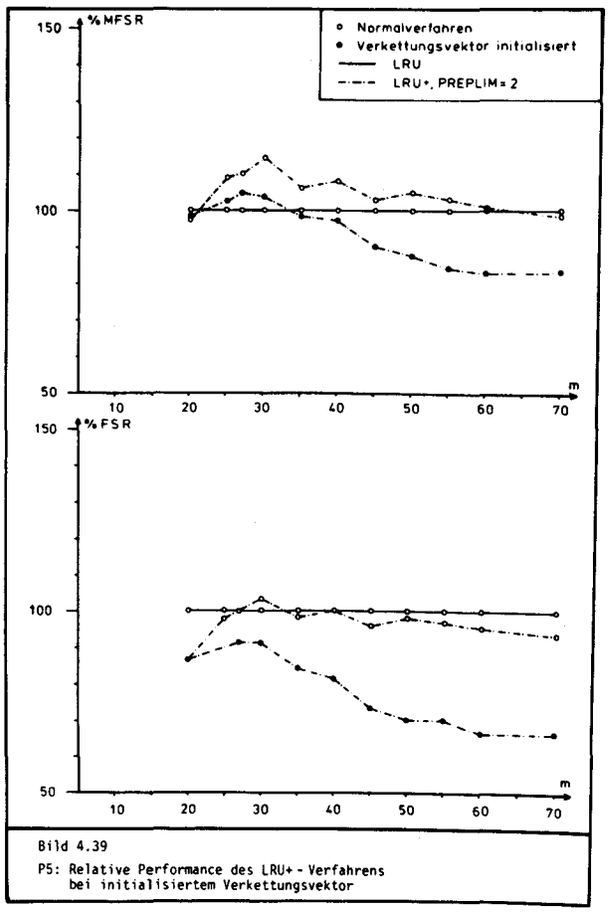


Bild 4.39
P5: Relative Performance des LRU+ - Verfahrens bei initialisiertem Verkettungsvektor

In-Raten zustande gekommen sind.

Bei diesen Messungen waren die Prepaging Faktoren bei großen Arbeitsspeichern deutlich vergrößert, was auf die Tatsache zurückzuführen ist, daß bereits bei leerem Arbeitsspeicher Prepaging betrieben werden kann und dann die größeren Verkettungslängen zum Tragen kommen, was die in Kap. 4.3.1 aufgestellte These erhärtet, daß bei Normalverfahren für große Speichergrößen der Prepaging Faktor deshalb relativ klein ist, weil sich mit hoher Wahrscheinlichkeit erhebliche Teile der benötigten Lokalitäten bereits im Arbeitsspeicher befinden. Der Effekt starken anfänglichen Prepagings wurde noch dadurch verstärkt, daß, um einen möglichst starken 'Preload'-Effekt zu erzielen, die Beschränkungen durch den PREPLIM-Parameter beim initialen Page Fault außer Kraft gesetzt waren.

Der Erfolg bei diesen Messungen zeigt, daß es wünschenswert ist, die während eines

Programmlaufs gesammelten Informationen über das Programmverhalten zu konservieren, um sie bei einem erneuten Programmaufruf verwerten zu können, weil dadurch, gerade im Bereich für die Praxis akzeptabler Fehlseitenraten erhebliche Performance-Verbesserungen möglich sind. Es sollte zum Schluß noch darauf hingewiesen werden, daß die relativen Auswirkungen dieser Maßnahme nicht nur mit steigender Fehlseitenrate, sondern auch mit steigender Programmlänge abnehmen.

4.3.4 Einfluß der Seitengröße

Die Seitengröße wirkt sich die Seitengröße in vielerlei Hinsicht auf die Performance von Programmen und Verfahren aus; es ist deshalb nicht leicht, losgelöst von den Gesamtwirkungen Aussagen über den Einfluß auf das Prepaging Verhalten zu gewinnen.

Zur Durchführung der Untersuchungen wurde ein Teil der Messungen unter Verwendung einer Seitengröße von 256 Bytes (d. h., $\frac{1}{8}$ der normalen Seitengröße von 2048 Bytes) wiederholt. Die Programmgrößen sind in Tab. 4.1 (Kap. 4.1) angegeben; sie sind aufgrund der dynamischen Fragmentierung deutlich kleiner als bei Verwendung großer Seiten (die Faktoren bei der Anzahl der von den Programmen belegten Seiten liegen zwischen 5 und 6, also deutlich unter 8). Zusätzlich zu den bekannten Auswirkungen einer Seitenverkleinerung bei Demand Paging Verfahren, die bei Prepaging Verfahren in gleicher Weise wirksam werden, sind für das Prepaging speziell zwei gegensätzlich wirkende Faktoren von Bedeutung:

(a) Da die Verkettungszeiten unabhängig von der Seitengröße sind, steigt die mittlere Länge der Verkettungen (was die Anzahl der verketteten Seiten, nicht aber was die Größe des durch Verkettungen verbundenen Speicherraumes angeht). Dies bedeutet - wie schon mehrfach dargelegt - eine Erhöhung des Prepaging Risikos (insbesondere bei unlimitiertem Prepaging.

(b) Durch die bei kleinen Seiten präzisere Modellierung der Lokalitäten sinkt die Wahrscheinlichkeit nicht eindeutiger Verkettungen, soweit sie nicht durch die

Programmlogik bedingt sind, und damit steigt die Zuverlässigkeit der Verkettungsinformation und das Prepaging Risiko verringert sich entsprechend.

Bei den Messungen wurden als alternative Vorgaben für Parameter PREPLIM die Werte 2 und 50 gewählt; sie sind insofern den bei den Messungen mit großen Seiten verwendeten Standardwerten 2 und 10 vergleichbar als 2 minimales und 50 (durch PREPLIM) unlimitiertes Prepaging bedeutet; letztere Aussage wurde durch Messungen mit PREPLIM = 100, die nur noch kleine Veränderungen ergaben, erhärtet.
In Bild 4.40 ist für P1 die Fehlseitenrate des LRU-Verfahrens und der davon abgeleiteten Prepaging Verfahren für beide Seitengrößen eingezeichnet. Auffällig ist, daß der extreme Anstieg der Fehlseitenrate beim Basisverfahren (der sich außerhalb des dargestellten Bereiches in mehreren Stufen fortsetzt) ebenso wie durch den Übergang zu Prepaging auch durch den Übergang zu kleineren Seiten nachhaltig entschärft wird. Ebenso ungewöhnlich ist, daß sich die Kurven der Fehlseitenraten der beiden Seitengrößen schneiden, und zwar schon bei relativ kleinen Werten der Fehlseitenrate, die bei den Basisverfahren kaum das Doppelte der primären Faults ausmachen; dies hat seine Ursache darin, daß die bei kleinen Seiten bessere Modellierung der Lokalitäten und die dadurch gegebene Möglichkeit, bei gleicher Speichergröße mehrere Lokalitäten bzw. Lokalitäten höherer Stufe im Arbeitsspeicher zu halten, sich bei Programm P1 außergewöhnlich stark auswirkt.

Die starken Auswirkungen der Seitenverkleinerung bereits auf das Basisverfahren lassen P1 nicht besonders geeignet erscheinen, den Einfluß der Seitengröße auf das Prepaging Verhalten zu studieren. Bild 4.40 zeigt jedoch, daß auch bei den kleinen Seiten die Prepaging Verfahren zu einer deutlichen Verbesserung der Performance gegenüber dem Basisverfahren führen.

Als nächstes soll kurz der Einfluß der Seitengröße beim DWS-Verfahren diskutiert werden (Bild 4.41). Ein vergleichbarer Einfluß der Seitengröße wie beim LRU-Verfahren ist beim DWS-Verfahren nicht feststellbar. Der bereits in Kap. 4.3.1

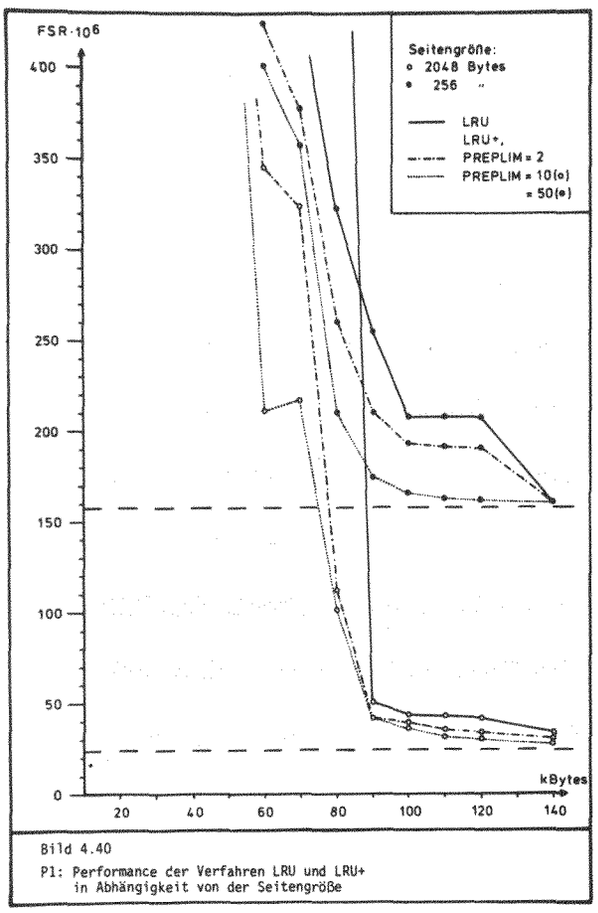


Bild 4.40
P1: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße

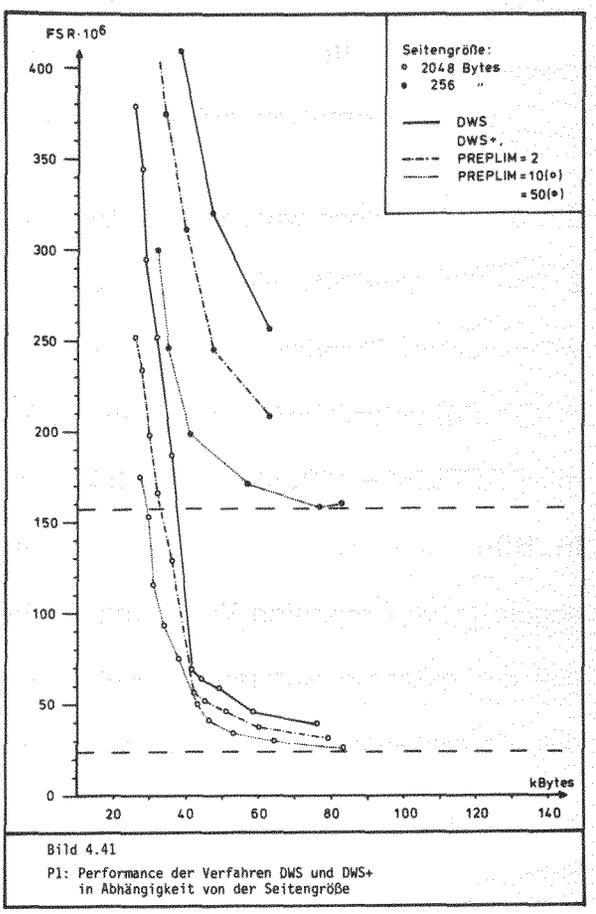


Bild 4.41
P1: Performance der Verfahren DWS und DWS+ in Abhängigkeit von der Seitengröße

erwähnte große Performance-Unterschied zwischen LRU- und DWS-Verfahren wird durch die kleine Seitengröße verringert, weil diese zu Performance-Verbesserungen beim LRU-Verfahren und Performance-Verschlechterungen beim DWS-Verfahren führt. Die Erhöhung der Fehlseitenrate bei Verkleinerung der Seiten bleibt bei allen Meßpunkten - auch außerhalb des dargestellten Bereiches - erhalten. Die Zahl der Programmseiten beträgt bei der kleinen Seitengröße das 6,5-fache der Seitenzahl bei großen Seiten. Um annähernd den gleichen Faktor (die Werte schwanken zwischen 5,7 und 10) steigt die Anzahl der Fehlseitenbedingungen bei jeweils gleicher Fenstergröße T. Daß der Performance-Unterschied in Bild 4.41 nicht so groß ist, liegt daran, daß die Verkleinerung der Seiten bei festem T nicht nur zu einer Erhöhung der Fehlseitenrate, sondern auch zu einer deutlichen Verkleinerung der mittleren Working Set Größe führt, vor allem bei kleinen T-Werten. Abschließend kann festgestellt werden, daß sich die Verkleinerung der Seiten beim DWS-Verfahren erwartungsgemäß auswirkt und daß auch hier durch Prepaging deutliche Performance-Verbesserungen zu erzielen sind.

Die weiteren Untersuchungen der Auswirkungen der Seitengröße werden an den Programmen P4 und P5 durchgeführt. In den Bildern 4.42 und 4.43 sind für P4 und P5 die Fehlseitenraten für das LRU-Verfahren und die abgeleiteten Prepaging Verfahren für beide Seitengrößen eingetragen. Bei beiden Programmen sind nennenswerte Veränderungen der Performance zwischen den jeweiligen Basisverfahren und Prepaging Verfahren der verschiedenen Seitengrößen nicht erkennbar. Bei P5 ist feststellbar, daß bei beiden Seitengrößen in gleicher Weise ein Performance-Gewinn nicht zu erzielen ist und daß verstärktes Prepaging eher zu einer Verschlechterung der Performance führt.

Bei P4 ist ebenso wie schon bei P1 erkennbar, daß die Abstände der Performance-Kurven voneinander bei kleinen Seiten vergrößert sind gegenüber den Abständen bei großen Seiten, der Performance-Gewinn durch Prepaging also vergrößert ist. Bei unlimitiertem Prepaging ist dies ganz deutlich, bei PREPLIM = 2 zumindest bei P4

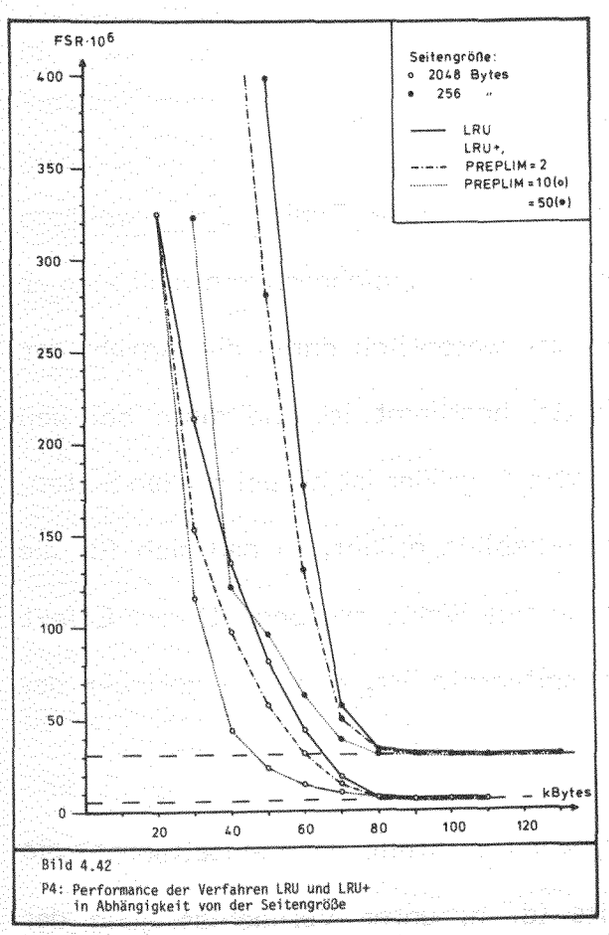


Bild 4.42
P4: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße

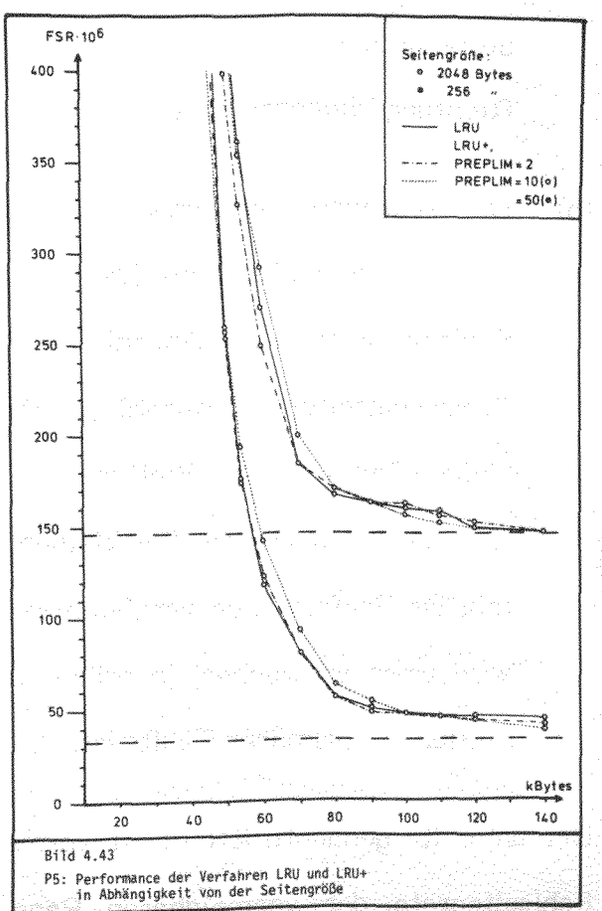


Bild 4.43
P5: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße

nicht; dies liegt daran, daß P4 ein für Prepaging gut geeignetes Programm ist, das vergleichsweise hohe Prepaging Faktoren erreicht (vgl. Tab. 4.14 in Kap. 4.3.1), so daß PREPLIM = 2 zu einer echten Einschränkung des Prepaging und des damit verbundenen Nutzens führt und diese Restriktion sich bei kleinen Seiten stärker auswirkt als bei großen Seiten.

In Bild 4.44 ist die relative Performance für P4 aufgetragen. Hierbei zeigt es sich, daß die relative Performance bei kleinen Seiten im Vergleich zur relativen Performance bei großen Seiten nicht so günstig verläuft wie man es aufgrund der absoluten Darstellung in Bild 4.42 erwartet hätte. Dies hat zwei Ursachen:

(a) Aufgrund der dynamischen Fragmentierung benötigt ein Programm bei kleinen Seitengrößen weniger Speicherplatz und erreicht deshalb bereits bei kleineren Arbeitsspeichergrößen den Zustand, daß über die primären Page Faults hinaus kaum Faults auftreten und somit Verbesserungen der Fehlseitenrate nicht mehr möglich sind. Dies ist auch daran zu erkennen, daß die für die beiden Seitengrößen ähnlich aussehenden Kurven für die kleinen Seiten etwas in Richtung kleinerer Arbeitsspeichergrößen verschoben sind.

(b) Die relativen Performance-Werte sind auf die Page Faults des jeweiligen Basisverfahrens bezogen. Da nun im Bereich großer Speichergrößen (und kleiner Fehlseitenraten) die Anzahl der Page Faults wesentlich durch die Anzahl der Programmseiten (= Anzahl primärer Faults) bestimmt ist und diese bei den kleinen Seiten um ein Vielfaches (etwa Faktor 6) größer ist als bei großen Seiten, sind die Bezugsgrößen bei kleinen Seiten erheblich größer, so daß sich für die relative Performance vergleichsweise ungünstige Werte ergeben. Dieser Effekt wird umso schwächer, je größer die Fehlseitenrate ist, d. h., je geringer der Einfluß der primären Faults ist.

Der unter (b) genannte Effekt ist vermeidbar, wenn an Stelle der Fehlseitenrate die Fehlseitenrate der vermeidbaren Page Faults (d. h. unter Vernachlässigung der

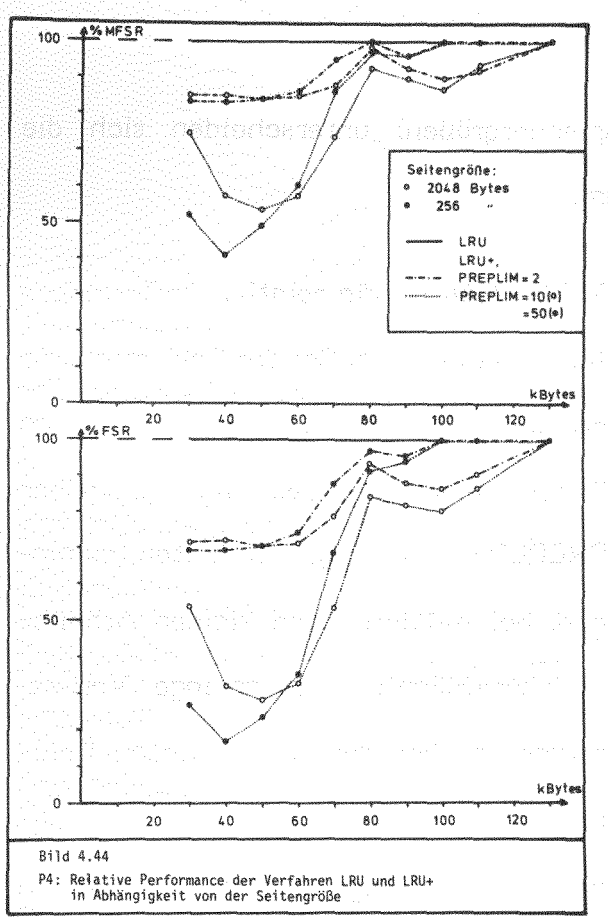


Bild 4.44

P4: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße

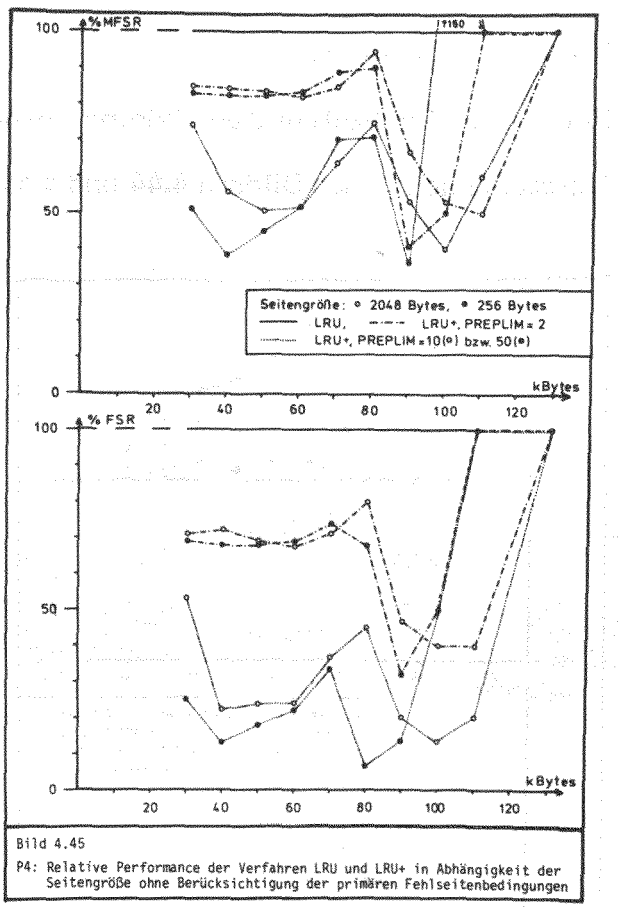


Bild 4.45

P4: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit der Seitengröße ohne Berücksichtigung der primären Fehlseitenbedingungen

primären Faults) verwendet wird. In diesem Fall wird die Anzahl der prinzipiell vermeidbaren Page Faults der Prepaging Verfahren zu der entsprechenden Zahl der Basisverfahren in Beziehung gesetzt. Dies ist für Programm P4 in Bild 4.45 geschehen. Daß der unter (a) beschriebene Effekt von dieser Änderung der Darstellung nicht berührt wird, ist klar und kann wie in Bild 4.44 aus der Verschiebung der Kurven für kleine Seiten in Richtung kleiner Speichergrößen abgelesen werden.

Diese Darstellung ist insofern eine interessante Alternative, als eine Beziehung zwischen der Performance der Verfahren und dem prinzipiell Machbaren hergestellt wird. Ihr Nachteil - und deshalb wird sie auch sonst nicht verwendet - ist, daß die relative Bedeutung der einzelnen Page Faults mit abnehmender Fehlseitenrate überproportional zunimmt.

Die Folge ist, daß bereits kleine, zufällige Verschiebungen der gegenseitigen Performance einen deutlichen Niederschlag finden, was solchen Effekten eine irreführend hohe Bedeutung zukommen läßt und überdies zu einem sehr unruhigen

Kurvenverlauf führt.

Für hohe Fehlseitenraten (kleine Arbeitsspeichergrößen) unterscheiden sich die Darstellungen in den Bildern 4.44 und 4.45 kaum.

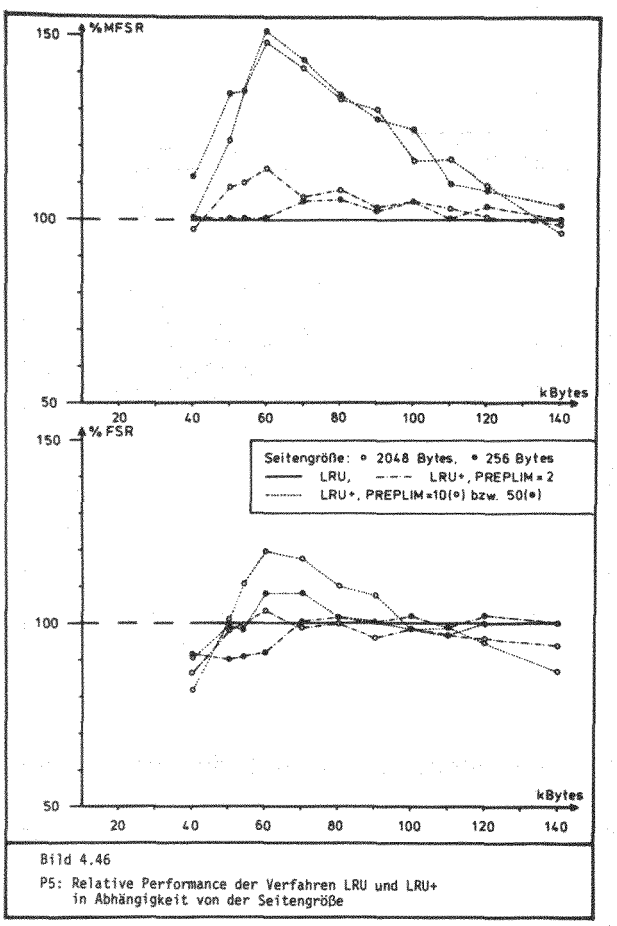


Bild 4.46
P5: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße

In Bild 4.46 ist die relative Performance für P5 dargestellt. Bei der Fehlseitenrate ergibt sich für PREPLIM = 50 bei kleinen Seiten eine Verbesserung gegenüber PREPLIM = 10 bei großen Seiten (zumindest bei mittleren und kleinen Arbeitsspeichergrößen); eine analoge Verbesserung ist bei der modifizierten Fehlseitenrate nicht feststellbar, was auf eine erhöhte Page In-Rate bei kleinen Seiten hindeutet. Bei PREPLIM = 2 schneiden die kleinen Seiten bei MFSR relativ etwas günstiger ab als bei FSR, eine Folge der sich bei kleinen Seiten stärker auswirkenden Limitierung beim Prepaging. Alle Unterschiede sind jedoch gering und ändern an den grundsätzlichen Verhältnissen zwischen Demand Paging Verfahren und Prepaging Verfahren nichts.

Zusammenfassend kann festgestellt werden, daß die Seitengröße keinen entscheidenden Einfluß auf die Funktion der Prepaging Verfahren hat. Verkleinerung der Seitengröße hat einen leicht positiven Effekt auf die Performance der Prepaging Verfahren in dem Sinne, daß die Performance durchweg geringfügig besser ist und die positiven Effekte des Prepaging etwas stabiler und gleichmäßiger sind. Es liegt auf der Hand, daß die in Kap. 4.3.3 beschriebenen durch Speicherung der Verkettungsinformation möglichen Effekte (Preload, sofortiges Prepaging) bei kleineren Seitengrößen noch verstärkt zum Tragen kommen.

4.3.5 Alternative Verkettungen

Es wurden Messungen mit den in Kap. 3.3.2.2 beschriebenen vereinfachten Verkettungsalgorithmen, im folgenden mit V1 und V2 bezeichnet, durchgeführt. Für das LRU-Verfahren sind die Ergebnisse für die Programme P1, P4 und P5 in den Bildern 4.47, 4.48 und 4.49 zusammen mit den Ergebnissen der Normalverkettung eingetragen; auch das normale LRU-Verfahren ist zum Vergleich eingezeichnet.
Zur Erleichterung der Interpretation der Bilder soll zunächst eine globale vergleichende Wertung der verschiedenen Verkettungen vorgenommen werden. Dazu ist in Tab. 4.15 die relative Performance der verschiedenen Varianten bezogen auf das Basis-LRU-Verfahren und gemittelt über alle Meßpunkte eingetragen. Um eine Überbewertung der Verhältnisse bei hohen Fehlseitenraten zu vermeiden, wurde dazu die relative Performance für jeden Meßpunkt ermittelt und anschließend von den Werten das Mittel gebildet.

Zunächst kann wertungsfrei festgestellt werden, daß beim Performance-Maß FSR die Verkettungsvariante V2 bei P1 geringfügig, bei P4 und P5 deutlich schlechter als die beiden Konkurrenten abschneidet und daß die Unterschiede beim Performance-Maß MFSR noch deutlicher sind.

Programm	LRU	LRU+, PREPLIM=2						LRU+, PREPLIM=10					
		FSR			MFSR			FSR			MFSR		
		normal	V1	V2	normal	V1	V2	normal	V1	V2	normal	V1	V2
P1	100	68,7	68,1	68,8	73,8	73,9	74,7	61,0	63,6	61,6	73,4	76,6	76,1
P4	100	83,9	83,2	100,4	91,0	90,6	111,1	67,2	78,1	90,9	80,0	88,2	102,2
P5	100	96,9	101,7	130,1	104,9	112,5	136,8	102,3	112,2	129,4	122,5	144,2	179,9

Tab. 4.15

Relative Performance der auf den Verkettungsvarianten basierenden Prepaging Verfahren bezogen auf das Normalverfahren LRU

Diese Beobachtung deckt sich mit dem, was aufgrund der Definition auch theoretisch ableitbar ist:

Bei der Verkettung V2 entfällt die Zeitlimitierung der Verkettungen und damit der Versuch, Verkettungen nur innerhalb von Lokalitäten vorzunehmen. Die Folge ist, daß die Programmseiten in einem sehr viel höheren Maße verkettet werden und die Unterschiede um so größer werden, je kleiner bei den Alternativverfahren die Verkettungszeit vorgewählt wurde, d. h., je kleiner die Arbeitsspeichergröße ist, da beim LRU-Verfahren Speichergröße und Verkettungszeit gekoppelt sind. Durch die erhöhte Zahl von Verkettungen kann in stärkerem Umfang Prepaging betrieben werden, wobei das Risiko über das normale Maß hinaus ansteigt, weil auch Verkettungen zwischen verschiedenen Programmteilen bestehen können. Das erhöhte Maß an Prepaging kann bei günstigem Programmverhalten, wie es etwa das Programm P1 zeigt - und dies ist auch den Bildern 4.47 bis 4.49 zu entnehmen - zu einem relativ günstigen Verhalten bei großen Speichergrößen führen, insbesondere auch dann, wenn das Risiko

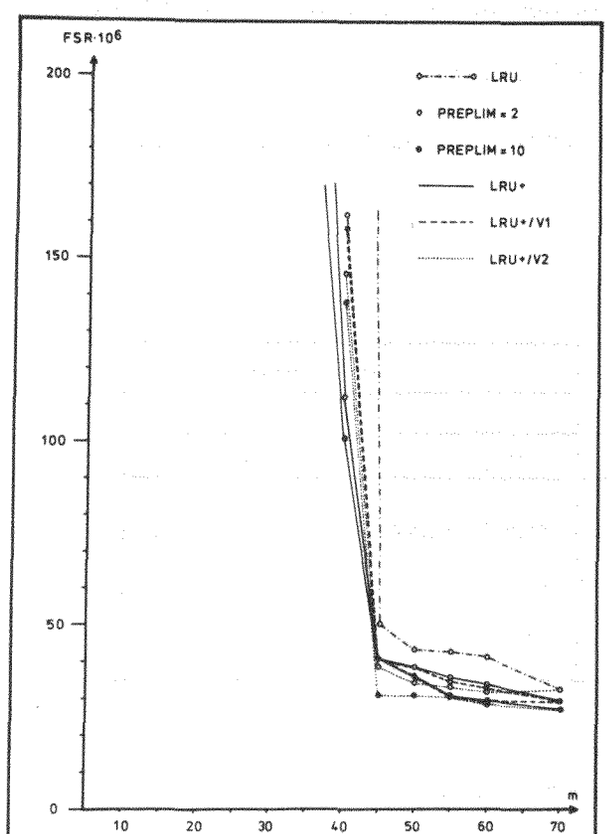


Bild 4.47
P1: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung

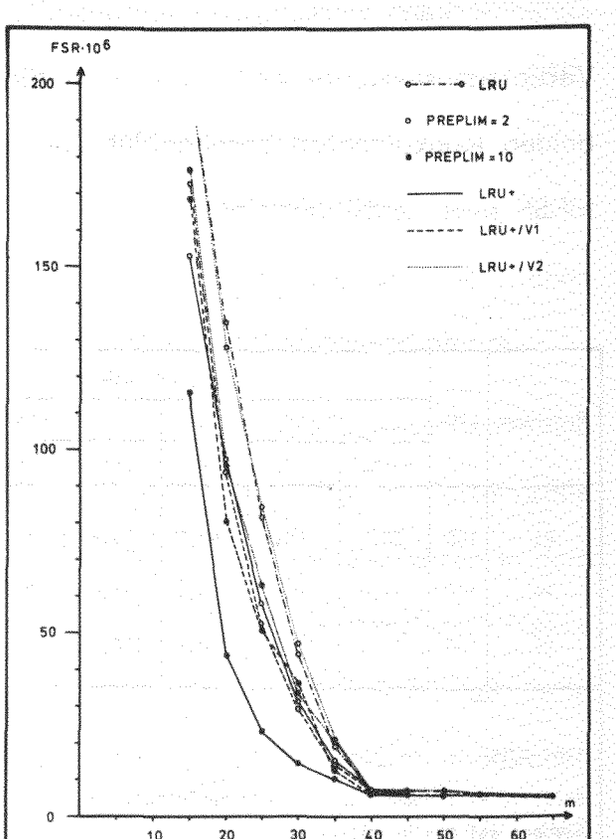


Bild 4.48
P4: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung

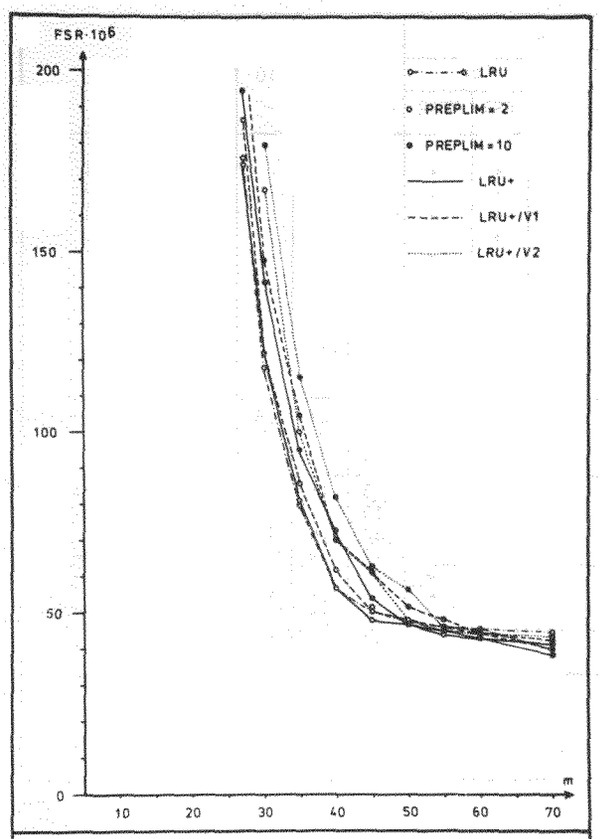


Bild 4.49
P5: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung

einer erhöhten Page In-Rate durch eine kleine Vorgabe von PREPLIM begrenzt ist. Bei kleinen Speichergrößen kommen die Nachteile des einfachen Verfahrens dann zum Tragen. Mit abnehmender Speichergröße wächst das Risiko beim Prepaging, weil der relative Anteil des verfügbaren Speichers, der für Prepaging beansprucht wird, steigt, was unter anderem dazu führt, daß zunehmend hochwertige Seiten im Sinne der LRU-Prioritätsliste aus dem Speicher entfernt werden müssen, um Prepaging betreiben zu können. Wenn unter diesen Umständen mit geringwertigen Verkettungsinformationen und evtl. in erheblichem Umfang (großes PREPLIM) Prepaging betrieben wird, dann sind deutliche Abstriche gegenüber den konkurrierenden Verfahren unvermeidlich. Die Daten in Tab. 4.15 und der Kurvenverlauf unterstreichen dies und lassen die Variante V2 im Vergleich zu V1 abfallen, so daß im folgenden nur noch die Variante V1 diskutiert wird.

Zur vergleichenden Wertung der beiden Verkettungen (Normalverkettung und V1) wird zunächst Tab. 4.16 herangezogen. In dieser Tabelle ist ein '+' - Zeichen eingetragen, wenn die Performance des Prepaging Verfahrens basierend auf der Normalverkettung besser ist als die des auf der Verkettung V1 basierenden Verfahrens. Da bei dieser Tabelle die Größe der Qualitätsunterschiede nicht erfaßt wird, kann daraus keine umfassende qualitative Wertung abgeleitet werden (so ist z. B. für MFSR mit PREPLIM = 2 bei P1 an vier Meßpunkten V1 besser als das Normalverfahren und nur zweimal das Normalverfahren besser; dennoch schneidet insgesamt das Normalverfah-

Index der Speicher-vorgabe	PREPLIM=2						PREPLIM=10						∑+	∑-	∑0	
	P1		P4		P5		P1		P4		P5					
	FSR	MFSR	FSR	MFSR	FSR	MFSR	FSR	MFSR	FSR	MFSR	FSR	MFSR				
1	+	+	+	+	+	+	+	+	-	+	+	+				'+' = Normal besser als V1
2	-	-	+	+	+	+	+	+	+	+	+	+				
3	-	-	-	-	+	+	-	+	+	+	+	+				
4	+	+	-	-	0	+	+	+	+	+	+	+				
5	-	+	-	-	+	+	0	-	+	+	+	+				
6	-	+	-	-	+	+	-	-	+	+	+	+				
7	-	-	0	0	+	+	-	+	+	+	+	+				
8	-	-	0	0	+	+	0	+	0	0	+	+				
9	0	+	0	0	+	+	-	+	0	0	+	+				
10	-	-	0	0	+	+	0	0	0	0	+	+				
11	0	0	0	0			0	0	0	0						
∑+	2	5	2	2	9	10	3	7	6	7	10	10	73	27	28	Gesamt
∑-	7	5	4	4	0	0	4	2	1	0	0	0				
∑0	2	1	5	5	1	0	4	2	4	4	0	0				
	*		*		*								13	11	8	FSR, PREPLIM=2
							*		*		*		19	5	8	FSR, PREPLIM=10
	*		*		*		*		*		*		32	16	16	FSR
		*		*		*							17	9	6	MFSR,PREPLIM=2
								*		*		*	24	2	6	MFSR,PREPLIM=10
		*		*		*		*		*		*	41	11	12	MFSR
	*	*	*	*	*	*							30	20	14	PREPLIM=2
							*	*	*	*	*	*	43	7	14	PREPLIM=10

Tab. 4.16

Relative Performance der LRU+ - Verfahren auf der Basis der Normalverkettung und der Variante V1

ren besser ab, wie aus Tab. 4.15 ersichtlich ist); es werden aber charakteristische Unterschiede sichtbar. Drei Dinge sind ablesbar:

(1) Beim Übergang vom Performance-Maß FSR zum Performance-Maß MFSR verschiebt sich die Relation zugunsten der Normalverkettung. Dies bedeutet, daß bei V1 tendenziell mehr Seiten geladen werden, wodurch unter Umständen jedoch auch eine Reduktion der Fehlseitenrate erzielt wird.

(2) Beim Übergang von PREPLIM = 2 zu PREPLIM = 10 erhöht sich das Risiko bei V1 stärker als bei der Normalverkettung, so daß sich eine Verschiebung zugunsten des Normalverfahrens ergibt.

(3) Vergleicht man die Gesamtzahl der '+' - und '-' - Zeichen (73: 27), so ergibt sich über alles eine Überlegenheit des Normalverfahrens.

Die Tendenzen, die in (1) und (2) sichtbar werden, decken sich mit dem, was man aufgrund der Definitionen der Verkettungen erwarten würde:

Die Normalverkettung ist eine restriktive Verkettung in dem Sinne, daß nicht sicher erscheinende Informationen (manifestiert dadurch, daß die Page Fault Folge abweicht von einer bereits eingetragenen Verkettung) gelöscht werden, wodurch insgesamt der Grad der Verkettungen verkleinert und insbesondere die mittlere Länge bestehender Verkettungen reduziert wird, d.h., die Menge der benutzbaren Informationen wird verkleinert, die Zuverlässigkeit der verbleibenden Verkettungen erhöht. Bei der Nutzung der Informationen wirkt sich das so aus, daß weniger Prepaging betrieben wird und die großen Risiken, die durch lange Verkettungen entstehen, abgebaut werden. Dies wirkt sich in der Performance positiv aus, wenn die Verkettungsinformationen unzuverlässig sind (wie es für nicht rekursive Programme typisch ist) oder wenn das Prepaging Risiko besonders groß ist, was bei langen Verkettungen und nicht durch PREPLIM eingeschränktem Prepaging und / oder kleinen Arbeitsspeichergrößen der Fall ist.

Bei der Verkettung V1 wird die Wertminderung der Verkettungsinformation, die aus der Tatsache der Veränderung resultiert, ignoriert und die neue Verkettung wertgleich eingetragen und nachfolgend benutzt. Beim Normalverfahren werden an solchen Stellen, um das Risiko unnötiger Ladevorgänge zu mindern, Page Faults in Kauf genommen. Sollte die neue Verkettung Bestand haben, bringt die Vorgehensweise beim V1-Verfahren Performance-Vorteile. Wird das allgemeine Prepaging Risiko durch Vorgabe nicht zu großer PREPLIM-Werte limitiert (was nach den Ausführungen in Kap. 4.3.1 ohnedies sinnvoll ist), so ist das Folgeseitenverfahren auf der Basis der Verkettung V1 für realistische Speichergrößen dem Normalverfahren fast ebenbürtig, insbesondere, wenn der geringere Implementationsaufwand und Overhead berücksichtigt werden. Noch geringer werden die Unterschiede, wenn durch programmabhängige Vorgabe von PREPLIM das vergleichsweise schlechte Abschneiden bei im Sinne des Prepaging schlecht konditionierten Programmen verhindert wird.

Die entsprechenden Vergleichsmessungen wurden auch für das DWS-Verfahren durch-

geführt. Sie führten ebenfalls zu dem Ergebnis, daß das Folgeseitenverfahren auf der Basis der Verkettung V1 für vernünftige Fenstergrößen T, d. h. für nicht zu kleine Arbeitsspeichergrößen $\bar{m}$ als im wesentlichen gleichwertiges Verfahren angesehen werden kann.

4.4 Zusammenfassung

Es wurden an fünf nach unterschiedlichen Kriterien ausgewählten Testprogrammen umfangreiche Meßreihen zur Feststellung der relativen Performance von Prepaging Verfahren und vergleichbaren Demand Paging Verfahren durchgeführt. Bei den optimalen Verfahren fester und variabler Arbeitsspeichergröße sind die Performance-Daten der Prepaging Verfahren in gleicher Weise für alle Testprogramme und über das gesamte Spektrum der Parameterwerte mit deutlichem Abstand besser als die der Demand Paging Verfahren. Diese Aussage gilt auch bei Verwendung des Performance-Maßes MFSR (vgl. Kap. 3.1.3), was die prinzipielle Überlegenheit der Prepaging Verfahren unterstreicht.

Bei den realisierbaren Verfahren wurden Folgeseiten-Verfahren auf der Basis des LRU-, DWS- und PFF-Verfahrens mit den jeweiligen Basisverfahren verglichen, wobei für die Untersuchung verschiedener spezieller Aspekte vorwiegend die Verfahren LRU und LRU+ herangezogen wurden, die als Verfahren fester Arbeitsspeichergröße für vergleichende Untersuchungen besonders geeignet sind.

Zunächst zeigen die Messungen, daß bei den Programmen P1 bis P4 die Fehlseitenrate der Prepaging Verfahren für alle Parametervorgaben stets kleiner ist als bei den Demand Paging Verfahren; diese Aussage bleibt qualitativ auch bei Verwendung der modifizierten Fehlseitenrate erhalten. Es bestätigt sich aber auch, daß - in Übereinstimmung mit Überlegungen in Kap. 3.3.2.1 - bei Programmen, die kein rekursives Verhalten in dem dort angegebenen Sinne zeigen und wofür Programm P5 ein extremes Beispiel ist, durch Prepaging nach dem Folgeseiten-Prinzip keine Performance-Verbesserungen möglich sind.

Als bemerkenswerte Eigenschaft der Prepaging Verfahren muß festgehalten werden, daß ihr positives Abschneiden beim Vergleich nicht an eng begrenzte Randbedingungen geknüpft ist. Dies manifestiert sich in der Eindeutigkeit der Relation bei allen Folgeseiten-Verfahren über das gesamte Spektrum der Parametervorgaben, und es wird

noch dadurch unterstrichen, daß auch die Veränderung von anderen, auf die Performance von Seitenaustauschverfahren wirksamen Größen, wie etwa eine Verkleinerung der Seitengröße, an den Verhältnissen nichts ändert. Darüber hinaus ist eine weitere Verschiebung zugunsten der Prepaging Verfahren möglich, wenn es gelingt, die am Ende eines Programmdurchlaufs in Form von Verkettungen vorhandene Information über das Programmverhalten zu konservieren und bei einem erneuten Aufruf des gleichen Programms nutzbar zu machen. Desgleichen zeigen die Messungen, und dies ist auch aufgrund theoretischer Überlegungen einsichtig, daß sich rekursives Programmverhalten stärker auswirken kann, wenn die Laufzeiten länger sind; somit führen die relativ kurzen Zeiten realer Programmlaufzeit, wie sie der beträchtliche Aufwand der Simulationen nur zuläßt, eher zu einer Benachteiligung der Prepaging Verfahren.

Schließlich haben Untersuchungen mit alternativen Methoden zur Erstellung der Verkettungsinformation deutlich gemacht, daß auch auf der Basis von Verkettungen, die mit sehr geringem Overhead erzeugt werden können, praktisch gleichwertige Prepaging Verfahren mit geringfügig abweichender Charakteristik entstehen.

Zusammenfassend muß als Ergebnis der durchgeführten Messungen festgehalten werden, daß Prepaging nach dem Folgeseiten-Prinzip, insbesondere bei Programmen mit Strukturen, wie sie für langlaufende Programme typisch sind, deutliche und stabile Performance-Verbesserungen gegenüber den jeweils zugrunde liegenden Demand Paging Verfahren bewirkt.

5 AUSBLICK

Die Untersuchungen dieser Arbeit haben gezeigt, daß es möglich ist, in realisierbaren Verfahren Informationen über Referenzstrukturen von Programmen zu gewinnen, die für Prepaging Verfahren verwertbar sind. Es ist aber sicher, daß das Erkennen von Struktur in den Referenzketten dadurch erschwert ist, daß die Referenzketten eine Superposition veschiedener Referenzströme (Instruktionen, skalare Konstanten und Variable und Datenfelder) darstellen, wobei für jede dieser Teilketten Struktur leichter sichtbar wird. Aus diesem Grunde und weil gerade bei Datenfeldern wegen der im allgemeinen einfachen Speichervorschrift und den häufig sehr einfachen Zugriffsalgorithmen die Referenzstruktur relativ einfach und damit leicht erkennbar und beschreibbar ist, beziehen sich die Untersuchungen in einer Reihe von Arbeiten /M 1, S 5, T 2/ nur auf Datenfelder. Mit der gleichen Begründung können auch Folgeseiten-Verfahren definiert werden, bei denen nur Referenzen auf Datenfelder oder Referenzen auf Datenfelder und Referenzen auf Instruktionen (einschließlich der skalaren Größen) getrennt verkettet werden. Man kann erwarten, daß dadurch eine zuverlässigere Beschreibung des zukünftigen Programmverhaltens durch die Verkettungsinformationen erreicht wird, weil nicht eindeutige Verkettungen entfallen, die durch Anwendung des gleichen Programmes auf verschiedene Datenfelder oder die Verarbeitung eines Datenfeldes durch verschiedene Programme zustande kommen.

Für eine ausschließliche Berücksichtigung der Datenfelder spricht auch die Entwicklung auf dem Hardware-Gebiet. Bereits seit einigen Jahren besteht in zunehmendem Maße die Tendenz, zugunsten eines größeren Arbeitsspeicherausbaus auf Trommel- bzw. Festkopfplattenspeicher zu verzichten, und in näherer Zukunft werden mit Arbeitsspeichergrößen von mehreren MBytes bei durchschnittlichen Systemen und deutlich über 10 MBytes bei großen Installationen die vorgenannten elektromechanischen Speicher vollständig überflüssig werden. Bezogen auf solche Arbeitsspeichergrößen werden Instruktionen und skalare Größen der überwiegenden Mehrzahl aller

Programme nur einen sehr kleinen Teil des verfügbaren Speicherplatzes beanspruchen, so daß man daran denken kann, das Paging auf Datenfelder zu beschränken.

Was die Auswirkungen der möglichen Entwicklungen der Hardware auf das Paging-Konzept angeht, so zeichnen sich zwei Alternativen ab:

(1) Trommel- und Festkopfplattenspeicher fallen als erste Stufe der Sekundärspeicher ersatzlos weg.

Wegen des dann größeren Geschwindigkeitsunterschiedes der direkt Information austauschenden Speichermedien (Arbeitsspeicher/Plattenspeicher mit beweglichem Arm) wird nach CHU und OPDERBECK /C 4/ die Verwendung größerer Seiten nahegelegt, so daß die Phänomene einer Seitenvergrößerung und relativ geringer Fehlseitenraten in den Vordergrund rücken.

(2) Trommel- und Festkopfplattenspeicher werden durch neue, leistungsfähigere vollelektronische Speichermedien ersetzt.

In diesem Falle werden höhere Fehlseitenraten möglich; dadurch ist nach CHU und OPDERBECK /C 4/ eine Verkleinerung der Seiten angebracht. Bei Systemen mit hohen Paging-Raten (etwa Timesharing-Systemen) hat bereits heute die durch die große Anzahl der Systemunterbrechungen verursachte Systembelastung ein Maß erreicht hat, das sinnvollerweise nicht wesentlich überschritten werden kann. Ein höherer Datenfluß zwischen den Speicherebenen ohne Erhöhung des System-Overheads kann auf zwei Arten erfolgen:

(a) Durch Verwendung einfachster Algorithmen wird der Overhead je Transaktion verringert.

Die Verwendung sehr einfacher Algorithmen würde es ermöglichen und nahelegen, die Steuerung des Informationsaustausches in die Hardware zu verlegen, wodurch eine gegenüber einer Software-Lösung sehr viel größere Reduktion des Overheads erreicht würde.

(b) Durch Verwendung von Prepaging Verfahren kann ein höherer Datenfluß bei gleichzeitiger Verringerung der Anzahl der Programmunterbrechungen erreicht werden.

Die Untersuchungen dieser Arbeit haben gezeigt, daß Prepaging Verfahren in der Lage sind, die Anzahl der Fehlseitenbedingungen und damit bei Verwendung autonomer E/A-Prozessoren die Anzahl der Systemunterbrechungen zu verringern, wobei dies aber in der Regel mit einer Erhöhung der Belastung des Paging-Kanals verbunden ist.

Zusammenfassend kann bezüglich der Entwicklung der Rechner-Hardware festgestellt werden, daß - bei allen Unsicherheiten im einzelnen - in der näheren Zukunft die Arbeitsspeichergrößen deutlich schneller als die Rechenleistung zunehmen werden, was bei herkömmlichen Anwendungen zu einer gewissen Entspannung der Paging Problematik führen wird; bei neueren Anwendungen (z. B. Datenbanksystemen) dürfte das Paging Problem jedoch kaum an Bedeutung verlieren.

Zum Schluß soll noch die Frage erörtert werden, in welche Richtung sich Prepaging Verfahren weiter entwickeln könnten. Es gibt drei Möglichkeiten der Realisierung von Prepaging Verfahren, die sich vor allem durch die Art und Weise der Beschaffung der für Prepaging verwertbaren Informationen über das Programmverhalten unterscheiden.

(1) Während des Programmablaufs werden durch das Betriebssystem aktuelle Informationen über das Programmverhalten gesammelt und nachfolgend noch während der Programmausführung verwertet. Diese Methode wurde bei den in dieser Arbeit vorgestellten Folgeseiten-Verfahren angewendet.

(2) Auf Grund von Modellvorstellungen des Programmverhaltens bzw. der Referenzstruktur werden Prepaging Algorithmen definiert; dies ist die Vorgehensweise in /S 5/.

(3) Informationen über das Programmverhalten liegen bereits vor Beginn der Pro-

grammausführung auf Grund von Angaben des Programmierers und/oder des Compilers vor. Diese Methode wurde von TRIVEDI /T 2/ untersucht.

Es ist festzustellen, daß die Methoden (2) und (3) ausschließlich auf Datenfelder angewendet wurden und auch nur dort Erfolg haben können, was jedoch nach den bisherigen Ausführungen in diesem Kapitel kein gravierender Nachteil sein muß. Interessant für zukünftige Untersuchungen könnten Prepaging Verfahren sein, die auf einer Kombination der Methoden (1) und (3) beruhen. Auf der Problemebene sind in erheblichem Umfang Kenntnisse über Referenzschemata von Datenfeldern vorhanden, die bisher auf dem Weg bis zum ausführbaren Maschinenprogramm verloren gehen. Es sollte prinzipiell möglich sein, solche Kenntnisse durch geeignete Sprachkonstrukte im Zusammenspiel mit darauf abgestimmten Betriebssystemen nutzbar zu machen. Um den Aufwand an dieser Stelle nicht allzu sehr anwachsen zu lassen, werden in der Regel nur bedingte Zusammenhänge herstellbar sein, die aber durch Kopplung an das Referenzgeschehen während der Programmausführung zu sicheren Aussagen über zukünftiges Programmverhalten führen.

LITERATUR

/A1/ Aho, A.V., Denning, P.J., Ullman, J.D., "Principles of Optimal Page Replacement", JACM, 18, 1, January, 1971, pp. 80-93

/A2/ Arora, R. K., Subramanian, R. K., "An Optimal Demand Prepaging Algorithm", Information Processing Letters, 7, 3, 1978, pp. 132-136

/B1/ Badel, M., Gelenbe, E., Lenfant, J., Leroudier, J., Potier, D., "Adaptive Optimization of the Performance of a Virtual Memory Computer", from Computer Architectures and Networks, ed. Gelenbe and Mahl, North Holland, 1974, pp. 1-25, republished, Proc. IEEE, 63, 6, June, 1975, pp. 958-965

/B2/ Baer, J. L., Sager, G. R., "Dynamic Improvement of Locality in Virtual Memory Systems", IEEE Trans. on Softw. Eng., SE-2, 1976, pp. 54-62

/B3/ Batson, A.P., Blatt, W.E., Kearns, J.P., "Structure Within Locality Invtervals", Proc. Symp. on Modeling and Performance Evaluation of Computer Systems, ed. Beilner and Gelenbe, North-Holland, October, 1977, pp. 221-232

/B4/ Belady, L.A., "A Study of Replacement Algorithms for a Virtual Storage Computer", IBM Systems J., 5, 2, 1966, pp. 78-101

/B5/ Belady, L.A., Kuehner, C.J., "Dynamic Space Sharing in Computer Systems", CACM, 12, 5, May, 1969, pp. 282-288

/B6/ Belady, L. A., Nelson, R. A. and Shedler, G. S., "An Anomaly in the Space-Time Characteristics of Certain Programs Running in Paging Machines", CACM 12, 6, June 1969, pp. 349-353

/B7/ Budzinski, R.G., "Dynamic Memory Allocation in a Virtual Memory System", Ph. D. Thesis, University of Illinois and Urbana, 1977

/C1/ Casey, R.G., Osman, I., "Replacement Algorithms for Storage Management and Relational Data Bases", The Computer J., 19, 4, November, 1976, pp. 306-314

/C2/ Chow, We-Min, Chiu, W.W., "A Program Behaviour Model for Paging Systems", IBM Research Report RC 6452, March, 1977, republished Proc. Int. Symp. Computer Performance Modeling, Measurement and Evaluation, Yorktown Heights, N.Y., August, 1977, pp. 363-380

/C3/ Chu, W., Opderbeck, H., "The Page Fault Frequency Replacement Algorithm", Proc. FJCC, 1972, 597-609

/C4/ Chu, W., Opderbeck, H., "Performance of Replacement Algorithms with Different Page Sizes", Computer, 7, 11, November, 1974, pp. 14-21

/C5/ Chu. W., Opderbeck, H., "Analysis of the PFF Replacement Algorithm via a Semi Markov Model", CACM, 19, 5, May, 1976, pp. 298-304

/C6/ Coffman Jr., E.G., Denning, P.J., "Operating Systems Theory", Prentice Hall, 1973

/C7/ Coffman, Jr., E.G., Ryan Jr., T.A., "A Study of Storage Partitioning Using a Mathematical Model of Locality", CACM, March, 1972, 15, 3, pp. 185-190

/D1/ Denning, P.J., "Resource Allocation in Multiprocess Computer Systems", (Ph.D. Thesis), MIT Project MAC Tech. Report MAC-TR-50, May, 1968

/D2/ Denning, P.J., "The Working Set Model for Program Behavior", CACM, 11, 5, May, 1968, pp. 323-333. (See also, Arthur Bernstein, "Comment on the Working Set Model for Program Behavior", CACM, 13, 11, November, 1970, pp. 698-699.)

/D3/ Denning, P.J., "Virtual Memory", Computer Surveys, 2, 3, September, 1970, pp. 153-189

/D4/ Denning, P.J., Graham, G.S., "Multiprogrammed Memory Management", Proc. IEEE, 63, 6, June, 1975, pp. 924-931

/D5/ Denning, P.J., Kahn, K., "A Study of Program Locality and Lifetime Functions", Proc. Fifth SIGOPS, Austin, Texas, November, 1975, pp. 207-216

/D6/ Denning, P.J., Schwartz, S.C., "Properties of the Working Set Model", CACM, 15, 3, March, 1972, pp. 191-198 (Corrigendum, CACM, 16, 2, February, 1973, pp. 122.)

/D7/ Denning, P.J., Slutz, D.R., "Generalized Working Sets for Segment Reference Strings", CACM, 21, 9, September, 1978

/F1/ Fagin, R., "A Counter-Intuitive Example of Computer Paging", CACM, 19, 2, February, 1976, pp. 96-97

/F2/ Ferrari, D., "Improving Program Locality by Strategy Oriented Restructuring", Proc. IFIP Conf., 1974, pp. 266-270

/F3/ Ferrari, D., "Improving Program Locality by Critical Working Sets", CACM, 17, 11, November, 1974, pp. 614-620

/F4/ Ferrari, D., "Tailoring Programs to Models of Program Behavior", IBM J. Res. and Devel., 19, 3, May, 1975, pp. 244-251

/F5/ Ferrari, D., Kobayashi, M., "Program Restructuring Algorithms for Global LRU Environments", Proc. Int. Computing Symp., 1977, Liege, Belgium, April, 1977, pp. 277-283

/F6/ Franklin, M.A., Graham, G.S., Gupta, R.K., "Anomalies With Variable Partition Paging Algorithms", CACM, 21, 3, March, 1978, pp. 232-236

/G1/ Gelenbe, E., "The Distribution of a Program in Primary and Fast Buffer Storage", CACM, 16, 7, July, 1973, pp. 431-434

/G2/ Ghanem, M.Z., "Dynamic Partitioning of Main Memory Using the Working Set Concept", IBM J. Res. and Devel., 19, 5, September, 1975, pp. 445-450

/G3/ Graham, G.S., "A Study of Program and Memory Policy Behavior", Ph. D. Thesis, Purdue University, 1976

/G4/ Gupta, R.K., "Program reference behavior and dynamic memory management", D. Sc. Diss., Dept EE. Washington U., St. Louis, Mo, 1974

/H1/ Hansen, P.B., "Operating Systems Principles", Prentice-Hall, 1973

/H2/ Hatfield, D.J., Gerald, J., "Program Restructuring for Virtual Memory", IBM Sys. J., 10, 3, 1971, pp. 168-193

/H3/ Hatfield, D.J., "Experiments on Page Size, Program Access Patterns and Virtual Memory Performance", IBM J. Res. and Devel., 16, 1, January, 1972, pp. 58-66

/H4/ Hoßfeld, F., Mertens, B., "Computing in der Großforschung, Ziele und Struktur des Jülicher Systems", Informatik-Fachberichte 15, Organisation von Rechenzentren, Springer 1978

/I1/ IBM Time Sharing System: System Logic Summary, GY28-2009

/I2/ Innes, D.R., "Exploiting the Least Recently Used Page Replacement Algorithm", Software - Practice and Experience, 7, 2, 1977, pp. 271-273

/K1/ Kain, R.Y., "How to Evaluate Page Replacement Algorithms", Operating Systems Review, 9, 5, (also Proc. Fifth SIGOPS, November, 1975, University of Texas at Austin), pp. 1-5

/K2/ Kleinrock, L., "Queuing Systems", Vol. 1, Wiley, 1976

/K3/ Krzesinski, A., Teunissen, P., "An Analysis of the Page Size Problem Using a Network Analyzer", Proc. "Symp. on Modeling and Performance Evaluation of Computer Systems", ed. Beilner and Gelenbe, North Holland, Oct. 1977, pp. 239-250

/K4/ Krzesinski, A., Teunissen, P., "A Multiclass Network Model of a Demand Paging Computer System", Acta Informatica 9, pp. 331-343 (1978)

/L1/ Liu, M.H., "A Study of Memory Management Algorithms and Performance Measures", Ph.D. Dissertation, University of California, Berkeley, 1976

/M1/ Madison, A.W., Batson, A.P., "Characteristics of Program Localities", Proc. Fifth SIGOPS Conf., 1975, pp. 64-73, CACM, 19, 5, May, 1976, pp. 51-58

/M2/ Mattson, R.L., Gecsei, J., Slutz, D.R., Traiger, I.L., "Evaluation Techniques for Storage Hierarchies", IBM Sys. J., 9, 2, 1970, pp. 78-117

/M3/ Morris, J.P., "Demand Paging Through Utilization of Working Sets on the MANIAC II", CACM, 15, 10, October, 1972, pp. 867-872

/M4/ Morrisson, J.E., "User Program Performance in Virtual Storage Systems", IBM Sys. J., 12, 3, 1973, pp. 216-237

/M5/ Mortenson, J., "Computer Storage Hierarchy Models and Analysis", Ph.D. Thesis, Stanford University, 1976

/O1/ Opderbeck, H., Chu, W.W., "Performance of the Page Fault Frequency Replacement Algorithm in a Multiprogramming Environment", Proc. IFIPS Conf., Stockholm, Sweden, August, 1974, pp. 235-241

/O2/ Opderbeck, H., Chu, W.W., "The Renewal Model for Program Behavior", SIAM J. on Computing, 4, 3, September, 1975, pp. 356-374

/P1/ Potier, D., "Analysis of Demand Paging Policies with Swapped Working Sets", Proc. Sixth SIGOPS, November, 1977, pp. 125-131

/P2/ Prieve, B.G., "Using Page Residency to Select a Working Set Parameter", CACM, 16, 10, October, 1973, pp. 619-620

/P3/ Prieve, B.G., "A Page Partition Replacement Algorithm", Ph.D. Dissertation, University of California, Berkeley, 1974

/P4/ Prieve, B.G., Fabry, R.S., "VMIN - An Optimal Variable Space Replacement Algorithm", CACM, 19, 5, May, 1976, pp. 295-297

/R1/ Rafii, A., "Empirical Analytic Studies of Program Reference Behavior", SLAC Report 197, July, 1976, Stanford Linear Accelerator Center, Stanford, Ca.

/R2/ Rao, J.T., "Memory Use Estimator Function of a Program Executing in a Paging Environment", Proc. Second Texas Conf. on Computing Sys., University of Texas, Austin, Texas, November, 1973, pp. 15-1 - 15-7

/R3/ Ribeyre, P., Saintoyant, P.Y., "A Predictive Tool for the Improvement of Program Behaviour", Proc. Int. Comp. Symp., Liege 1977, pp. 285-290

/R4/ Richter, L., "Betriebssysteme", Teubner Studienbücher Informatik, Bd. 29, 1977

/R5/ Rodriguez-Rosell, J., "Experimental Data on How Program Behavior Affects the Choice of Scheduling Parameters", Proc. Third SIGOPS, October, 1971, Stanford University, pp. 156-163

/R6/ Rodriquez-Rosell, J., "Empirical Working Set Behavior", CACM, 16, 9, September, 1973, pp. 556-560

/R7/ Rodriguez-Rosell, J., Dupuy, J.P., "The Design, Implementation and Evaluation of a Working Set Dispatcher", CACM, 16, 4, April, 1973, pp. 247-253

/S1/ Sadeh, E., "An Analysis of the Performance of a Page Fault Frequency Replacement Algorithm", Proc. Fifth SIGOPS, November, 1975, University of Texas, Austin, Texas, pp. 6-13

/S2/ Sayre, D., "Is Automatic Folding of Programs Efficient Enough to Displace Manual", CACM, 12, 12, December, 1969, pp. 656-660

/S3/ Slutz, D.R., Traiger, I.L., "A Note on the Calculation of Average Working Set Size", CACM, 17, 10, October, 1974, pp. 563-565

/S4/ Smith, A. J., "Bibliography on Paging and Related Topics", Operating Systems Review, 12, 4, October, 1978, pp. 39-56

/S5/ Spaniol, O., "Demand Prepaging Algorithms Basing on a Model of Locality of Programs", presented at Symp. on Computer Architectures and Networks, IRIA, August, 1974

/S6/ Spirn, J. R., Denning, P. J., Experiments with Program Locality, Proc. FJCC, 1972, pp. 611-621

/T1/ Thorington Jr., J.M., Irwin, J.D., "An Adaptive Replacement Algorithm for Paged Memory Computer Systems", IEEETC, C-21, 10, October, 1972, pp. 1053-1061

/T2/ Trivedi, K. S., "Prepaging and Applications to Array Algorithms", IEEE Trans. on Comp., C-25, 1976, pp. 915-921

/T3/ Trivedi, K. S., "An Analysis of Prepaging", Computing 22, 1979, pp. 191-210

/T4/ Tsichritzis, D.C., Bernstein, P.A., "Operating Systems", Academic Press, 1974

/T5/ Turner, R., Strecker, B., "Use of the LRU Stack Depth Distribution for Simulation of Paging Behavior", CACM, 20, 11, November, 1977, pp. 795-798

/V1/ Vantilborgh, H., "On the Working Set Size and its Normal Approximation", BIT 14 (1974), pp. 240-251

/V2/ Vantilborgh, H., "Working Set Dynamics", Proc. 'Symp. on Modeling and Performance Evaluation of Comp. Systems 1976', North Holland, pp. 377-387

/W1/ Weber, G., "Mittlere Gültigkeitsdauer von Speicherzuständen als Vergleichskriterium für Ersetzungs-Strategien in virtuellen Speicher-Systemen", Elektron. Rechenanlagen 18, 5, 1976, pp. 220-223

/Y1/ Yu, F.S., "Modeling the Write Behavior of Computer Programs", Ph.D. Thesis, Stanford University, 1976

Anhang A

Tabellen der Performance-Daten der optimalen Verfahren

	FSR · 10^6				MFSR · 10^6		Page In's pro 10^6 Referenzen
m	LRU	OPT	PREOPT	OPR	PREOPT	OPR	OPR
20	1887,6	594,7	561,6	127,8	577,9	351,9	969,0
30	821,0	259,4	235,5	29,9	245,6	102,3	350,6
35	816,7	157,4	131,8	25,0	144,2	68,7	188,6
40	742,9	56,8	31,5	20,7	42,3	36,7	65,1
45	50,0	33,9	10,5	1,9	18,9	8,4	37,4
50	43,3	32,0	7,8	1,3	15,8	6,9	35,2
55	42,5	30,7	6,7	1,1	14,4	6,0	33,1
60	40,9	29,3	4,0	1,1	10,9	5,8	31,2
70	32,8	26,6	2,2	0,8	7,6	4,9	29,6
80	30,7	24,2	0,5	0,5	3,6	3,6	24,2
90	24,2	24,2	0,3	0,3	2,6	2,6	24,2

Tab. 4.4

P1: Performance-Daten für LRU, OPT, PREOPT und OPR

	FSR · 10^6				MFSR · 10^6		Page In's pro 10^6 Referenzen
m	LRU	OPT	PREOPT	OPR	PREOPT	OPR	OPR
20	1925,6	598,3	566,3	130,1	582,1	355,6	972,1
30	817,3	255,9	234,3	29,8	244,9	101,9	348,2
35	813,0	153,4	132,0	25,0	142,9	67,9	184,5
40	749,5	51,6	31,6	20,8	40,4	35,4	60,4
45	46,1	27,4	9,8	1,7	16,4	7,4	32,3
50	38,5	24,1	7,3	1,2	13,3	5,6	27,3
55	36,7	21,9	5,2	1,1	10,7	5,3	25,7
60	34,2	19,6	3,3	0,9	8,1	4,4	21,9
70	22,4	15,1	1,6	0,7	4,9	3,5	16,9
80	20,4	11,1	1,2	0,5	3,6	2,5	11,9
90	8,6	8,6	0,4	0,3	1,8	1,5	8,6

Tab. 4.5

P2: Performance-Daten für LRU, OPT, PREOPT und OPR

	FSR · 10^6				MFSR · 10^6		Page In's pro 10^6 Referenzen
m	LRU	OPT	PREOPT	OPR	PREOPT	OPR	OPR
10	1457,8	771,0	542,2	337,7	646,5	579,6	994,8
12	553,0	389,5	164,4	70,8	253,1	184,1	478,9
15	410,0	203,3	157,2	62,7	178,8	135,7	298,5
18	260,0	76,3	59,1	43,0	67,2	63,3	93,0
20	60,7	26,9	21,7	10,5	24,2	20,5	40,3
23	27,1	8,5	4,2	3,6	6,0	5,8	9,4
25	6,9	5,0	1,0	0,5	2,2	1,7	5,8
30	5,3	4,2	0,7	0,5	1,7	1,6	4,9
35	4,6	3,9	0,3	0,3	1,1	1,1	4,0
40	3,9	3,9	0,1	0,1	0,6	0,6	3,9

Tab. 4.6

P3: Performance-Daten für LRU, OPT, PREOPT und OPR

	FSR · 10^6				MFSR · 10^6		Page In's pro 10^6 Referenzen
m	LRU	OPT	PREOPT	OPR	PREOPT	OPR	OPR
10	325,4	323,2	71,3	68,8	151,8	149,3	324,1
15	213,9	212,9	31,3	29,2	81,3	79,0	213,4
20	134,7	134,3	17,3	15,9	48,3	46,2	134,4
25	81,4	81,4	10,3	9,1	28,9	27,2	81,4
30	43,9	43,9	7,1	6,2	17,7	16,5	43,9
35	19,0	19,0	4,5	3,8	9,3	8,5	19,0
40	7,1	7,1	1,1	0,7	2,8	2,3	7,1
45	6,6	6,3	0,5	0,2	1,8	1,2	6,3
50	6,6	5,9	0,5	0,2	1,8	1,2	5,9
55	6,1	5,5	0,5	0,2	1,7	1,1	5,5
65	5,1	5,0	0,2	0,1	1,1	0,9	5,0

Tab. 4.7

P4: Performance-Daten für LRU, OPT, PREOPT und OPR

	FSR · 10^6				MFSR · 10^6		Page In's pro 10^6 Referenzen OPR
m	LRU	OPT	PREOPT	OPR	PREOPT	OPR	
20	901,9	455,6	287,5	161,2	361,9	311,6	602,3
25	258,1	138,4	73,7	40,5	101,0	83,7	173,0
27	175,9	100,7	49,8	24,4	70,8	55,4	125,9
30	118,0	73,7	32,9	12,3	49,3	33,8	92,9
35	81,1	51,0	22,0	6,6	33,5	20,5	63,0
40	56,6	42,4	11,6	3,8	22,2	13,9	50,7
45	50,0	39,3	7,6	2,6	17,3	11,2	48,1
50	47,4	37,0	7,1	1,9	16,2	9,2	44,8
55	46,0	35,1	6,6	1,4	15,3	7,7	42,2
60	45,5	33,9	5,2	1,2	13,3	7,0	41,0
70	44,3	32,9	3,6	0,9	10,8	5,9	36,5

Tab. 4.8

P5: Performance-Daten für LRU, OPT, PREOPT und OPR

T	DWS		VMIN		VMINPP1			VMINPP2		
	FSR	$\overline{m}$	FSR	$\overline{m}$	FSR	MFSR	$\overline{m}$	FSR	MFSR	$\overline{m}$
1000	1329,3	11,09	1329,3	9,77	325,6	657,9	10,27	348,2	617,6	10,44
5000	378,6	12,89	378,6	11,03	101,4	196,0	11,45	111,4	202,1	12,67
7500	345,2	13,77	345,2	11,23	80,7	166,9	11,79	68,6	151,6	13,30
10000	295,5	14,55	295,5	11,66	57,6	130,4	12,18	48,2	117,5	13,55
15000	251,6	15,88	251,6	12,22	40,1	100,4	12,94	40,6	99,7	13,91
25000	187,3	17,93	187,3	13,45	25,0	68,5	14,50	23,4	64,2	18,76
50000	70,0	20,74	70,0	17,69	7,0	22,1	18,95	6,46	20,3	24,44
75000	64,8	22,19	64,8	18,04	4,6	17,2	19,95	1,6	9,3	34,15
125000	59,2	24,64	59,2	18,58	2,7	12,6	22,27	0,8	6,4	34,94
250000	45,7	29,33	45,7	21,30	1,6	8,6	25,40	0,5	4,6	36,62
750000	39,3	38,26	39,3	23,87	0,8	5,6	31,23	0,5	4,6	36,51

Tab. 4.9

P1: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2

T	DWS		VMIN		VMINPP1			VMINPP2		
	FSR	$\bar{m}$	FSR	$\bar{m}$	FSR	MFSR	$\bar{m}$	FSR	MFSR	$\bar{m}$
1000	1358,0	11,11	1358,0	9,76	335,6	675,1	10,27	357,3	632,5	10,44
5000	378,7	12,94	378,7	11,06	102,4	197,0	11,47	111,5	202,3	12,69
7500	347,4	13,83	347,4	11,24	81,3	168,1	11,80	70,0	153,6	13,29
10000	299,0	14,63	299,0	11,67	58,4	132,2	12,18	48,5	118,6	13,57
15000	255,0	16,01	255,0	12,22	40,7	101,9	12,95	41,3	101,3	13,91
25000	183,1	18,14	181,1	13,63	25,1	67,8	14,69	23,8	63,9	18,97
50000	64,8	20,96	64,8	17,87	6,5	20,6	19,14	6,1	18,9	24,51
75000	60,1	22,43	60,1	18,16	4,3	16,1	20,09	1,3	8,2	33,83
125000	55,8	25,11	55,8	18,59	2,6	12,0	22,12	0,7	5,7	34,90
250000	42,1	30,75	42,1	21,36	1,4	7,8	25,85	0,5	4,5	35,17
750000	32,4	44,42	32,4	25,22	0,6	4,5	30,60	0,5	4,2	32,36

Tab. 4.10

P2: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2

T	DWS		VMIN		VMINPP1			VMINPP2		
	FSR	$\overline{m}$	FSR	$\overline{m}$	FSR	MFSR	$\overline{m}$	FSR	MFSR	$\overline{m}$
1000	1436,6	6,68	1436,6	5,24	522,7	866,6	5,70	586,0	916,1	5,73
5000	248,8	8,82	248,8	7,59	51,7	113,4	7,66	52,7	114,5	7,65
7500	52,5	9,23	52,5	8,86	22,9	34,7	8,91	25,0	36,2	8,88
10000	44,0	9,35	44,0	8,93	17,8	28,0	9,00	22,2	31,3	8,95
15000	35,6	9,52	35,6	9,03	13,6	22,0	9,10	16,4	24,2	9,05
25000	28,9	9,81	28,9	9,16	11,3	18,0	9,22	11,5	18,3	9,21
50000	19,2	10,33	19,2	9,51	7,8	12,3	9,65	7,9	12,3	9,55
75000	14,6	10,68	14,6	9,80	6,1	9,5	9,95	6,3	9,6	9,83
125000	8,8	11,06	8,8	10,36	2,3	4,5	10,40	2,4	4,6	10,39
250000	8,0	11,67	8,0	10,52	1,4	3,3	10,59	1,4	3,3	10,59
750000	7,3	13,66	7,3	10,80	0,7	2,2	11,26	0,6	2,1	11,21

Tab. 4.11

P3: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2

T	DWS		VMIN		VMINPP1			VMINPP2		
	FSR	$\overline{m}$	FSR	$\overline{m}$	FSR	MFSR	$\overline{m}$	FSR	MFSR	$\overline{m}$
1000	464,8	5,51	464,8	5,05	459,5	462,1	5,05	460,1	462,3	5,05
5000	430,2	7,29	430,2	5,15	155,7	258,8	5,95	225,2	311,3	7,68
7500	411,6	8,34	411,6	5,26	105,3	208,1	6,56	30,3	110,8	13,81
10000	393,3	9,34	393,3	5,42	79,3	176,6	7,17	28,1	103,9	13,53
15000	359,0	11,21	359,0	5,85	56,8	142,8	8,16	24,0	90,6	13,45
25000	298,8	14,47	298,8	7,05	32,8	98,9	10,42	19,0	72,2	13,91
50000	183,8	20,34	183,8	11,27	14,8	52,1	15,38	11,4	43,5	18,04
75000	105,4	23,79	105,4	16,10	8,2	29,3	19,44	7,6	28,3	20,39
125000	23,6	26,51	23,6	23,99	4,3	10,0	24,60	4,9	10,7	24,53
250000	8,1	27,10	8,1	26,12	1,9	4,0	26,43	2,6	4,6	26,32
750000	7,7	28,62	7,7	26,30	1,1	2,9	28,29	1,0	2,8	28,16

Tab. 4.12

P4: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2

T	DWS		VMIN		VMINPP1			VMINPP2		
	FSR	$\overline{m}$	FSR	$\overline{m}$	FSR	MFSR	$\overline{m}$	FSR	MFSR	$\overline{m}$
1000	3086,3	13,69	3086,3	10,61	598,5	1359,1	11,60	595,2	1333,9	11,90
5000	598,5	18,37	598,5	15,40	119,0	266,9	16,28	130,6	278,5	16,20
7500	437,3	19,62	437,3	16,38	84,4	192,1	17,33	93,4	201,6	17,27
10000	351,7	20,58	351,7	17,12	63,5	149,5	18,21	70,6	156,9	18,15
15000	268,6	22,07	268,6	18,13	42,9	107,3	19,34	44,8	108,9	19,63
25000	197,0	24,26	197,0	19,50	27,7	73,9	21,20	27,3	72,2	21,67
50000	137,7	28,09	137,7	21,58	14,5	44,6	23,68	11,9	39,9	25,27
75000	105,2	30,83	105,2	23,54	10,0	32,4	25,86	9,2	30,7	26,97
125000	80,1	34,85	80,1	25,97	6,2	22,2	28,49	5,0	19,6	30,58
250000	55,7	41,30	55,7	30,04	3,1	13,1	33,39	2,1	10,8	35,89
750000	44,3	58,67	44,3	34,55	1,4	7,9	38,71	1,2	7,2	39,97

Tab. 4.13

P5: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2

Anhang B

Bezeichnungen, Definitionen, Begriffserläuterungen

Bevor einige der für das Fachgebiet und die Arbeit wichtigen Begriffe erläutert werden, sind einige Worte über die Bezeichnungen zu sagen. Auf dem Gebiet der praxisorientierten Betriebssystemlehre existiert praktisch keine deutschsprachige Originalliteratur; daher stammen sämtliche Begriffsbildungen aus dem Englischen; ein Teil der Begriffe ist inzwischen so verbreitet, daß er ohne Erklärung im Deutschen verwendet werden kann, für einen weiteren Teil existieren deutsche Übersetzungen und für den Rest, der vorwiegend aus speziellen und damit seltener auftretenden Begriffen besteht, existieren keine adäquaten deutschen Wortschöpfungen. Um den Einstieg in die Originalliteratur zu erleichtern, werden selbst in der einführenden deutschsprachigen Literatur (wie z. B. /R4/) fast immer zu den deutschen Begriffen auch die englischen Bezeichnungen genannt. Angesichts der aus dem vorher Gesagten folgenden Tatsache, daß jemand, der sich mit dem Fachgebiet bereits auseinandergesetzt hat, die englischen Begriffe ohnedies kennt und jemand, der sich damit befassen will, nicht umhinkommt, diese kennenzulernen, werden in dieser Arbeit englische und deutsche Begriffe zwanglos nebeneinander verwendet, nachdem bei der Einführung möglichst alle gängigen Bezeichnungen (englische und deutsche) genannt worden sind. Die deutschen Begriffe werden bevorzugt dann verwendet, wenn sonst englische Worte nach deutschem Sprachgebrauch dekliniert oder konjugiert werden müßten oder gemischtsprachige Wortkombinationen entstehen würden.

Die Begriffsbildungen und die Notation entsprechen im wesentlichen denen in /D4/ und /C6/. Die Begriffe, die nicht von übergeordneter Bedeutung sind, werden jeweils dort dort definiert, wo sie verwendet werden. Im folgenden werden einige grundlegende Definitionen und Beziehungen zwischen den definierten Größen angegeben.

Es bezeichnet

$N = \{1,\ldots, n\}$	die Menge der Seiten eines Programmes PR
$\omega = r_1 r_2 \ldots r_t \ldots$	die bei der Ausführung eines Programmes PR erzeugte Seitenreferenzfolge (auch Referenzkette),
$\|\omega\|$	deren Länge ($r_t = x$ bedeutet, daß die t - te Referenz auf eine Adresse der Seite x erfolgte)
M	die Menge der Seitenrahmen des realen Arbeitsspeichers
m	die Anzahl der für das Programm PR im Arbeitsspeicher zur Verfügung gestellten Seitenrahmen
$\overline{m}$	die im Mittel von PR belegte Anzahl der Seitenrahmen; es ist m, $\overline{m} \leq \|M\|$
A	einen Seitenaustauschalgorithmus
$\{S_t(A, m, \omega)\}$	(t = 0,1,...) die Folge von Speicherzuständen, die der Seitenaustauschalgorithmus A bei der Verarbeitung von ω in einem Speicher der Größe m Seiten erzeugt; S_0 ist ein gegebener Anfangszustand, im folgenden gilt grundsätzlich $S_0 = \emptyset$; S_t beschreibt die Untermenge von N, die sich nach der t- ten Referenz im Arbeitsspeicher befindet.

Es gilt

$$S_t \subseteq N, \quad |S_t| \leq m,$$

$$r_t \in S_t \quad \text{für } t > 0 .$$

Zwischen S_t und S_{t-1} besteht folgende Beziehung

$$S_t = S_{t-1} + X_t - Y_t,$$

wobei

$X_t \subseteq N - S_{t-1}$ die Menge der zum Zeitpunkt t zu ladenden Seiten und

$Y_t \subseteq S_{t-1}$ die Menge der zu diesem Zeitpunkt zu ersetzenden Seiten ist.

Zum Schluß sollen noch einige Begriffe erläutert werden, die in ihrer Bedeutung in der Literatur nicht immer eindeutig sind und die für das Verständnis der Arbeit von Bedeutung sind.

Activity Set (Aktivitätsmenge)

Die Aktivitätsmenge ist diejenige Teilmenge des Working Set (also der geschätzten Lokalität), die ein laufendes Programm aktuell benötigt. Die Bedeutung läßt sich am DWS-Verfahren sehr gut aufzeigen: Man betrachte ein laufendes Programm an der Nahtstelle zweier (der Einfachheit halber disjunkter) Programmphasen; dann gehören nach dem DWS-Verfahren die Seiten der beendeten Phase noch so lange zum Working Set, wie sie in das vorgegebene Zeitfenster fallen, sie gehören aber nicht zur aktuellen Aktivitätsmenge.

Lokalität

Der Begriff 'Lokalität' bezeichnet sowohl das Phänomen, daß Programme in Phasen ablaufen, denen bestimmte Eigenschaften zugeordnet werden können, als auch die Phase selbst als auch konkret die Menge der Seiten, die eine Programmphase ausmachen. An Stellen, wo eine Unterscheidung wichtig ist, wird das Phänomen als lokales (Programm-)Verhalten bezeichnet.

Performance

Unter Performance versteht man die quantitative und qualitative Leistungsfähigkeit, wobei letzteres in Richtung Güte und Qualität zu deuten ist. Der Begriff findet in vielen Bereichen der Datenverarbeitung sowohl

für Systeme als auch für Verfahren Anwendung.

Phasen-/Übergangsmodell (phase/transition model)

Dieser Begriff ist eng mit dem lokalen Programmverhaltens verknüpft. Bei neueren Modellen des Programmverhaltens wird die Ausführung eines Programmes als Folge von Phasen angesehen, wobei eine Phase eine Periode konstanter Speicheranforderung ist; eine Lokalität ist die Menge der Seiten, die während einer Phase referiert wird (ursprünglich die Menge der Programmsegmente, die in Paging Systemen jedoch in eindeutiger Weise Seiten zugeordnet sind). Es hat sich gezeigt, daß Programme an den Übergangsstellen zwischen aufeinander folgenden Phasen ein völlig unvorhersehbares Referenzverhalten an den Tag legen können, so daß diese bei einer Modellierung des Programmverhaltens unter Umständen einer gesonderten Behandlung bedürfen.

Prozeßzeit (virtuelle Zeit)

Die Prozeßzeit mißt diskret in Referenzen die reinen Ausführungszeiten eines Programmes.

Realzeit

Die Realzeit, die ebenfalls diskret in Referenzen gemessen wird, erfaßt zusätzlich zu den reinen Ausführungszeiten (= Prozeßzeit) vom Programm selbst verursachte Programmunterbrechungen, soweit sie durch Seitenwechselvorgänge verursacht sind (File - I/O bleibt unberücksichtigt!). In die durch Seitenwechselvorgänge verursachten Verzögerungen können auch systembedingte Wartezeiten eingebracht werden, wobei dann jedoch vorausgesetzt werden muß, daß sich das System in einem stationären Zustand befindet, weil sonst die Umgebungsunabhängigkeit der Zeitmessung, die für Performance-Vergleiche zwischen Seitenaustauschverfahren essentiell ist, nicht gegeben ist.

Working Set (Arbeitsmenge)

Allgemeiner versteht man unter einem Working Set (WS) eine durch ein Seitenaustauschverfahren geschätzte Lokalität (von Seiten); enger gefaßt bezeichnet der Begriff - weil er erstmals in diesem Zusammenhang verwendet wurde - die durch das von DENNING /D1/ vorgestellte Working Set (WS-)Verfahren vorgenommene Schätzung der Lokalität. In dieser Arbeit wird der Begriff in der allgemeineren Bedeutung verwendet; wenn der spezielle, durch das Denningsche Verfahren definierte Working Set gemeint ist, geht dies aus dem Text eindeutig hervor. Neuerdings - und auch in dieser Arbeit - wird zur Klarstellung das Denningsche Verfahren nicht mehr als WS-Verfahren, sondern als DWS-Verfahren bezeichnet.

ANHANG C

Verzeichnis der Tabellen und Bilder

TABELLEN

	Seite
Tab. 3.1 Simulationszeiten für die Verfahren PFF und PFF+	129
Tab. 3.2 P1: Simulationszeiten für die Verfahren LRU und LRU+	132
Tab. 4.1 Simulationsprogramme	135
Tab. 4.2 Standard-Parameterwerte beim DWS- und PFF-Verfahren	137
Tab. 4.3 Standard-Vorgaben der Arbeitsspeichergrößen bei Verfahren fester Speichergröße	138
Tab. 4.4 P1: Performance-Daten für LRU, OPT, PREOPT und OPR	193
Tab. 4.5 P2: Performance-Daten für LRU, OPT, PREOPT und OPR	194
Tab. 4.6 P3: Performance-Daten für LRU, OPT, PREOPT und OPR	195
Tab. 4.7 P4: Performance-Daten für LRU, OPT, PREOPT und OPR	196
Tab. 4.8 P5: Performance-Daten für LRU, OPT, PREOPT und OPR	197
Tab. 4.9 P1: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2	198
Tab. 4.10 P2: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2	199
Tab. 4.11 P3: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2	200
Tab. 4.12 P4: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2	201
Tab. 4.13 P5: Performance-Daten für DWS, VMIN, VMINPP1 und VMINPP2	202
Tab. 4.14 Prepaging Faktoren beim LRU+-Verfahren	158
Tab. 4.15 Relative Performance der auf den Verkettungsvarianten basierenden Prepaging Verfahren bezogen auf das Normalverfahren LRU	173

Tab. 4.16
Relative Performance der LRU+-Verfahren auf der Basis der Normalverkettung und der Variante V1 176

BILDER

	Seite
Bild 2.1 Fehlseitenrate als Funktion der Arbeitsspeichergröße	7
Bild 2.2 Mittlere Gültigkeitsdauer als Funktion der Arbeits-speichergröße	9
Bild 2.3 Space-time Produkt als Funktion der Arbeitsspeichergröße	11
Bild 2.4 Working Set des DWS-Verfahrens bei Fenstergröße T	20
Bild 2.5 Verhalten des DWS-Verfahrens an Phasenübergängen	22
Bild 2.6 Verhalten des PFF-Verfahrens an Phasenübergängen	26
Bild 2.7 Verhalten des VMIN-Verfahrens an Phasenübergängen	32
Bild 2.8 Grundsätzliche Konstellation für Ungültigkeit der Einschlußbedingung beim PFF-Verfahren	43
Bild 2.9 Beispiel für anomales Verhalten des PFF-Verfahrens	44
Bild 2.10 Beispiel für anomales Verhalten des MWS-Verfahrens	46
Bild 2.11 Beispiel für anomales Verhalten der LRU-Typ-Verfahren mit zeitgesteuertem UIC bei zufälliger Auswahl der zu ersetzenden Seite	47
Bild 2.12 Beispiel für anomales Verhalten des 2-Klassen LRU-Typ-Verfahrens bei zufälliger Auswahl der zu ersetzenden Seite	52
Bild 2.13 Beispiel für anomales Verhalten des 2-Klassen LRU-Typ-Verfahrens bei Anordnung der Seiten nach absteigenden Seitennummern	53
Bild 2.14 Berechnung des virtuellen STP beim DWS-Verfahren mittels Page Fault korrelierter Informationen	63

	Seite
Bild 3.1 Skizze zur Optimalität des PREOPT-Verfahrens	82
Bild 3.2 Skizze zur Regularität des OPR-Verfahrens	92
Bild 3.3 Ungültigkeit der Einschlußbedingung beim OPR-Verfahren	94
Bild 3.4 Das VMINPP1-Verfahren	97
Bild 3.5 Beispiel für anomales Verhalten des VMINPP1-Verfahrens	99
Bild 3.6 Bedeutung des Zeitparameters T bei Prepaging Verfahren	100
Bild 3.7 Das VMINPP2-Verfahren	102
Bild 3.8 Einschlußeigenschaft beim VMINPP2-Verfahren	105
Bild 3.9 Nicht eindeutige Verkettung	114
Bild 3.10 Flußdiagramm der Erstellung, Überprüfung und Korrektur von Verkettungen	116
Bild 3.11 Flußdiagramm der Erstellung und Korrektur von Verkettungen nach dem ersten vereinfachten Verfahren	118
Bild 4.1 P1: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens	140
Bild 4.2 P2: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens	140
Bild 4.3 P3: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens	140
Bild 4.4 P4: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens	140
Bild 4.5 P5: Performance des OPT-, PREOPT-, OPR- und LRU-Verfahrens	141

	Seite
Bild 4.6 P1: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens	142
Bild 4.7 P2: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens	142
Bild 4.8 P3: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens	142
Bild 4.9 P4: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens	142
Bild 4.10 P5: Performance des VMIN-, VMINPP1-, VMINPP2- und DWS-Verfahrens	143
Bild 4.11 P1: Performance der Verfahren LRU und LRU+	147
Bild 4.12 P2: Performance der Verfahren LRU und LRU+	147
Bild 4.13 P3: Performance der Verfahren LRU und LRU+	147
Bild 4.14 P4: Performance der Verfahren LRU und LRU+	147
Bild 4.15 P5: Performance der Verfahren LRU und LRU+	148
Bild 4.16 P1: Performance der Verfahren DWS und DWS+	149
Bild 4.17 P2: Performance der Verfahren DWS und DWS+	149
Bild 4.18 P3: Performance der Verfahren DWS und DWS+	149
Bild 4.19 P4: Performance der Verfahren DWS und DWS+	149
Bild 4.20 P5: Performance der Verfahren DWS und DWS+	150
Bild 4.21 P1: Performance der Verfahren PFF und PFF+	151
Bild 4.22 P2: Performance der Verfahren PFF und PFF+	151

Seite

Bild 4.23
P3: Performance der Verfahren PFF und PFF+ 151

Bild 4.24
P4: Performance der Verfahren PFF und PFF+ 151

Bild 4.25
P5: Performance der Verfahren PFF und PFF+ 152

Bild 4.26
P1: Relative Performance der Verfahren LRU und LRU+ 155

Bild 4.27
P2: Relative Performance der Verfahren LRU und LRU+ 155

Bild 4.28
P4: Relative Performance der Verfahren LRU und LRU+ 155

Bild 4.29
P5: Relative Performance der Verfahren LRU und LRU+ 155

Bild 4.30
P1: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM = 2/10 157

Bild 4.31
P2: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM = 2/10 157

Bild 4.32
P4: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM = 2/10 157

Bild 4.33
P5: Vergleich der relativen Performance FSR/MFSR bei den Verfahren LRU+, PREPLIM = 2/10 157

Bild 4.34
Vergleich der relativen Performance der Programme P1 und P2 für das Verfahren LRU+, PREPLIM = 2 162

Bild 4.35
Vergleich der relativen Performance der Programme P1 und P2 für das Verfahren LRU+, PREPLIM = 10 162

Bild 4.36
P1: Performance der Prepaging Verfahren LRU+ und DWS+ bei initialisiertem Verkettungsvektor 164

Bild 4.37
P5: Performance der Prepaging Verfahren LRU+ und DWS+ bei initialisiertem Verkettungsvektor 164

Bild 4.38
P1: Relative Performance des LRU+-Verfahrens bei initialisiertem Verkettungsvektor 165

Seite

Bild 4.39
P5: Relative Performance des LRU+-Verfahrens bei initialisiertem Verkettungsvektor 165

Bild 4.40
P1: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße 168

Bild 4.41
P1: Performance der Verfahren DWS und DWS+ in Abhängigkeit von der Seitengröße 168

Bild 4.42
P4: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße 169

Bild 4.43
P5: Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße 169

Bild 4.44
P4: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße 170

Bild 4.45
P4: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit der Seitengröße ohne Berücksichtigung der primären Fehlseitenbedingungen 170

Bild 4.46
P5: Relative Performance der Verfahren LRU und LRU+ in Abhängigkeit von der Seitengröße 172

Bild 4.47
P1: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung 174

Bild 4.48
P4: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung 174

Bild 4.49
P5: Performance des Verfahrens LRU+ für verschiedene Varianten der Verkettung 175