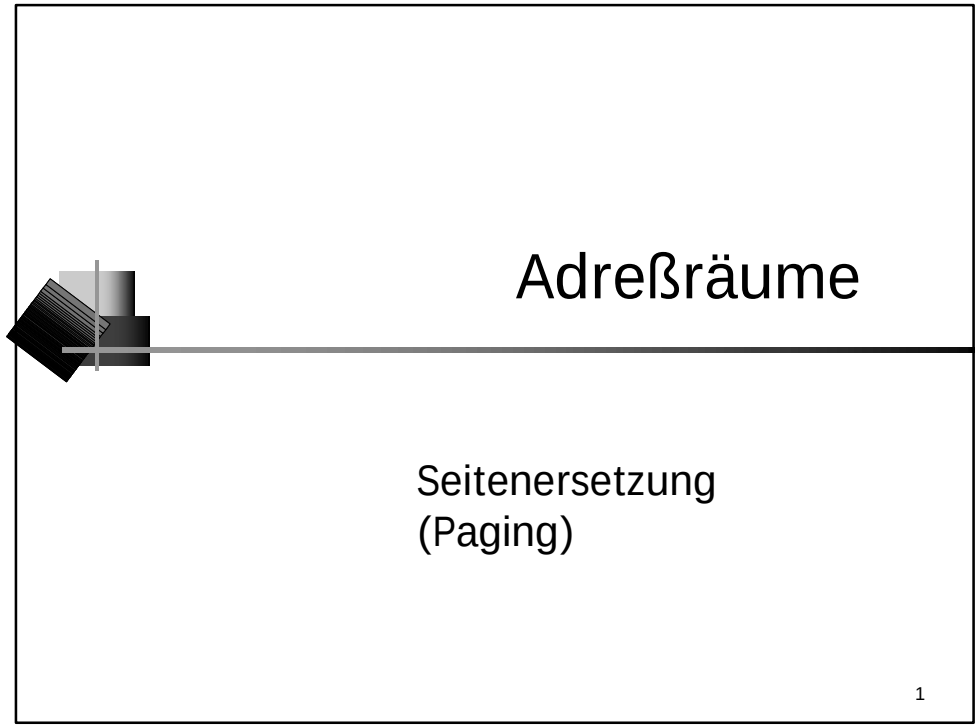
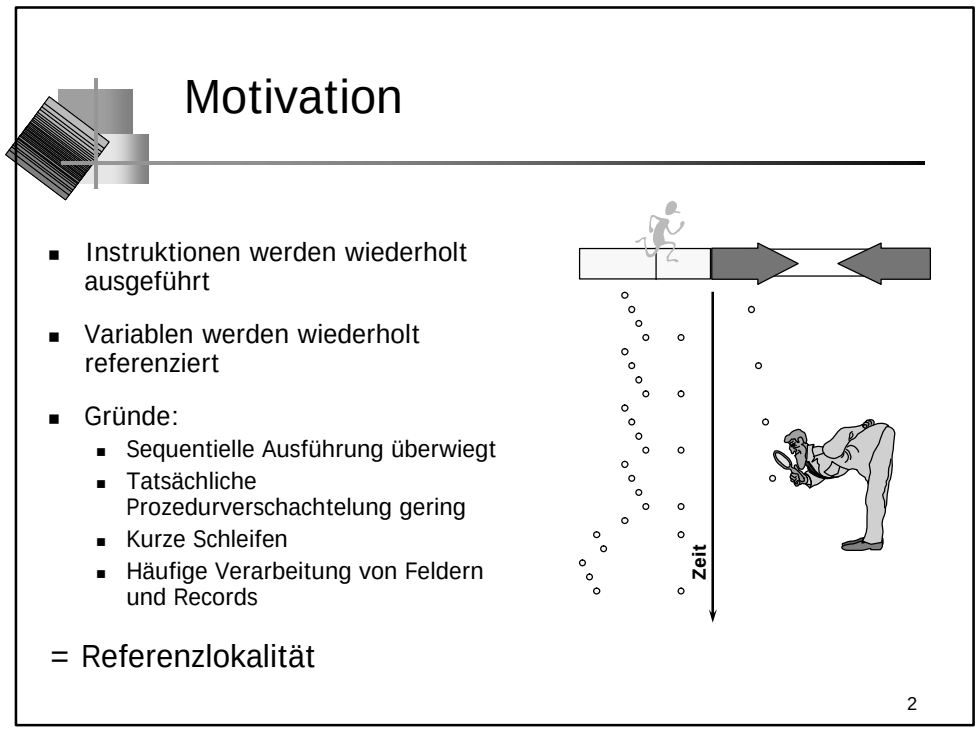
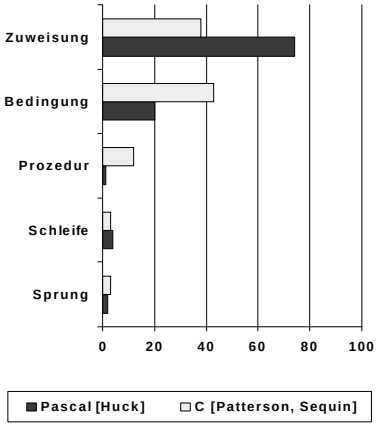

Adreßräume
Seitenersetzung
(Paging)
1


Motivation
Instruktionen werden wiederholt ausgeführt
Variablen werden wiederholt referenziert
Gründe:
Sequentielle Ausführung überwiegt
Tatsächliche Prozedurverschachtelung gering
Kurze Schleifen
Häufige Verarbeitung von Feldern und Records
= Referenzlokalität
Zeit
2

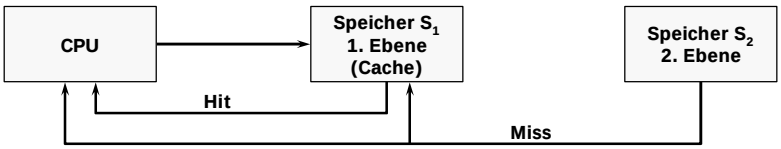
Erfahrungen mit Pascal und C

- Relative dynamische Frequenz von Hochsprachenkonstrukten im Quellcode
- Zuweisung und Bedingung ergeben über 80%
- Referenzen:
  - T. Huck
    Comparative Analysis of Computer Architectures
    Technical Report 83-243, Stanford University, Mai 1983
  - D. Patterson, C. Sequin
    A VLSI RISC
    IEEE Computer, Sept. 1982



3

Caching: Ausnutzung der Referenzlokalität



- 2-Ebenen-Speicher
- Kenngrößen eines Speichers S_k:
  - T_k = Mittlere Zugriffszeit
  - P_k = Preis pro Byte
  - G_k = Größe in Byte
- Trefferrate H
  - Wahrscheinlichkeit, daß referenzierter Wert im ersten Speicher ist

4

Mittlere Zugriffszeit und Kosten

- Mittlere Zugriffszeit:

$$T = HT_1 + (1-H)(T_1 + T_2) = T_1 + (1-H)T_2$$

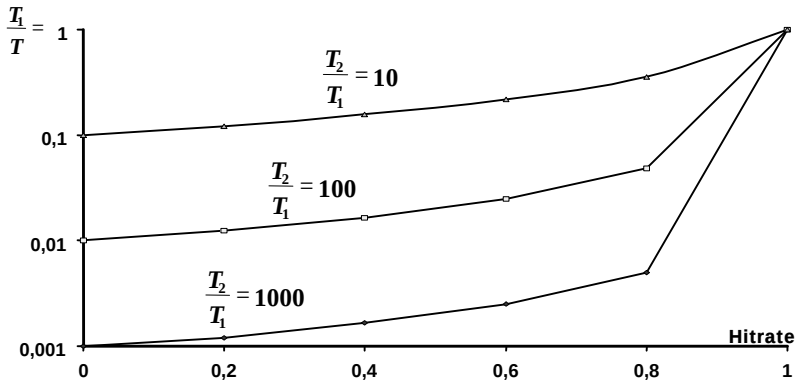
- Kosten:

$$P = \frac{P_1 G_1 + P_2 G_2}{G_1 + G_2}$$

- Zugriffseffizienz: $\frac{T_1}{T} = \frac{1}{H + (1-H)\frac{T_2}{T_1}}$

5

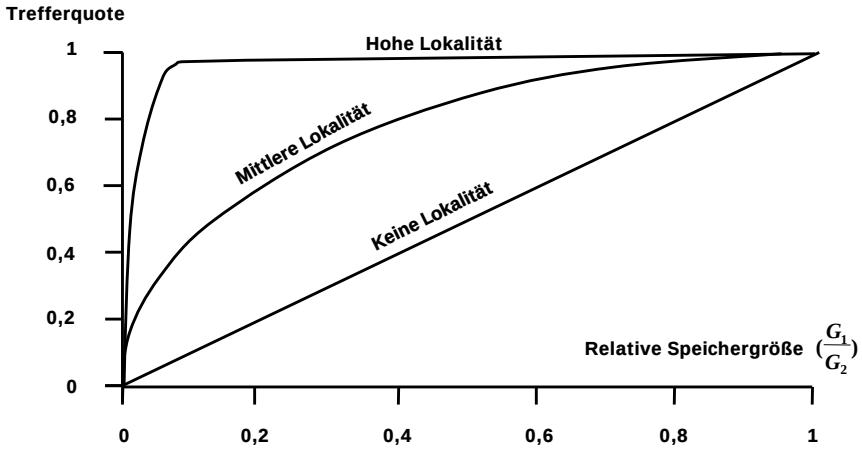
Zugriffseffizienz



- Typisch bei L2-Cache zu Hauptspeicher: $5 \le \frac{T_2}{T_1} \le 10$

6

Relative Cache-Größe



7

Optimierungsziel

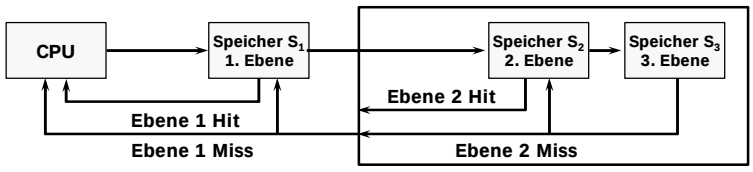
- Anstrebenswert ist:

$$T_1 << T_2: \quad T \approx T_1$$
$$C_2 << C_1: \quad C \approx C_2$$

- Erfordert Trefferquote H zwischen 0.8 und 0.9
- Bereits sehr kleine Caches ($G_1 << G_2$) liefern hohe Trefferquote

8

Caching über mehrere Ebenen

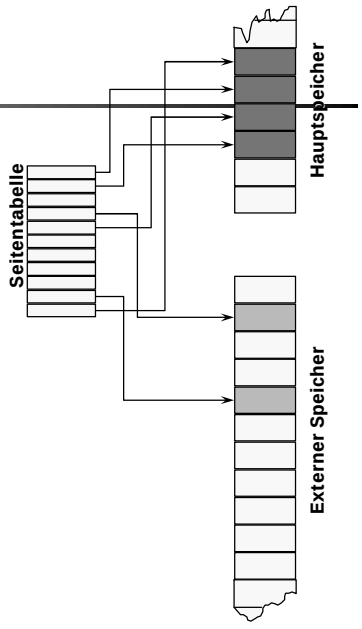


- Ausnutzung der Speicherhierarchie
  - L1-Cache und MMU-TLB zu L2-Cache
  - L2-Cache zu Hauptspeicher
  - Hauptspeicher
  - Externer Speicher
- Caching zwischen internem und externem Speicher besonders vielversprechend:
  - Verhältnis bei Zugriffszeit: $1:10^5$
  - Größenverhältnis: $1:10^2$-10^3
  - Kostenverhältnis: 1:2000

9

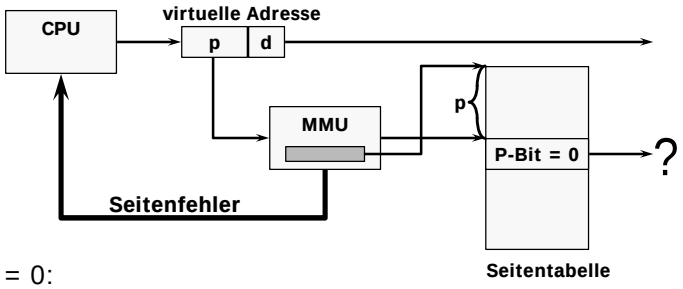
Paging

- Vorteile:
  - Vergrößerung des virtuellen Adreßraums
  - Adreßraum nur durch Größe des externen Speichers beschränkt
  - Gleichzeitige Unterstützung mehrerer Adreßräume
- Referenzlokalität verspricht minimale Zeitverzögerung?
- Ein- und Auslagern einzelner Seiten
  - P-Bit moderner MMU's
- Seitenfehler beim Zugriff auf ausgelagerte Seiten



10

Seitenfehler



- P-Bit = 0:
  - Seite ausgelagert
  - Adreßraum nicht angelegt
- Unterscheidung durch:
  - speziellen Adreßwert im Seitendeskriptor
  - Nutzung der freien Bits in den Deskriptoren

11

Behandlung von Seitenfehlern

1. Adreßraum nicht angelegt (ungültige Referenz)?
   - Programm terminieren

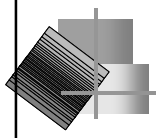
Seite ausgelagert!

2. Freie Kachel im Arbeitsspeicher suchen
   - Belegte Kachel verdrängen (Auslagern), wenn keine freie Kachel vorhanden
3. Seitentransfer von Platte einplanen

Kontextwechsel: Ausführung anderer Kontrollflüsse

4. Seitentabelle aktualisieren
   - P-Bit setzen
   - Adresse der gewählten Kachel eintragen
5. Instruktion wiederholen
   - Instruktionswiederholung muß von CPU unterstützt werden

12



Effizienz von Seitenersetzungsverfahren

- Effektive Zugriffszeit abhängig von der Wahrscheinlichkeit eines Seitenfehlers p:

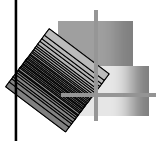
$$t_{effektiv} = (1-p)*t_{HS} + p*t_{SF}$$

  - t_{HS} = Zugriffszeit Hauptspeicher (ca. 70ns)
  - t_{SF} = Zeit für die Behandlung eines Seitenfehlers

- Seitenfehlerbehandlung:

$$t_{SF} = t_{Interrupt} + t_{Suchen} + t_{Auslagern} + t_{Einlagern} + t_{Wdh}$$
$$\approx 2*t_{Ein-/Auslagern}$$

13



Effektive Zugriffszeit

- Annahme:
  - Speicherzugriffszeit: 100 nsec
  - Plattenzugriff: 10 msec

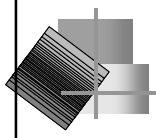
$$t_{effektiv} = (1-p)*10^{-7} + 2p*10^{-2}$$
$$\approx 2p*10^{-2}$$

- Weniger als 10% Zeitverzögerung:

$$1.1*10^{-7} > 10^{-7} + 2p*10^{-2}$$
$$p < 5*10^{-7}$$

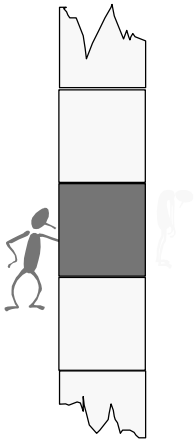
  - höchstens alle 2 Millionen Referenzen ein Seitenfehler

14

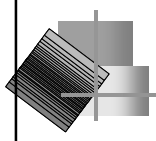


Seitenverdrängungsstrategien

- Seitenfehler: keine freie Kachel für einzulagernde Seite
- Konsequenzen:
  - Anwendung wird terminiert
  - ein anderer Adreßraum wird vollständig ausgelagert (Swapping)
  - eine bereits belegte Kachel wird frei gemacht
- Fall 3: Seitenverdrängung
- Welche Seite soll verdrängt werden?
  = Verdrängungsstrategie
  - minimale Seitenfehlerrate



15



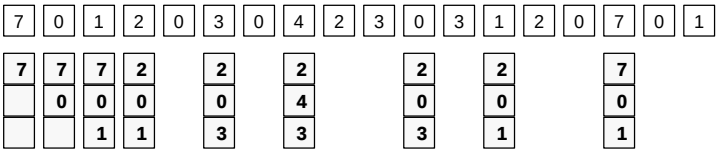
Referenzstring

- Bewertung von Verdrängungsstrategien
- Berechnet sich aus der Zugriffsfolge
  - bei welchen Zugriffen entstehen Seitenfehler?
  - nur Seitenanteil wichtig
  - Seite für eine bestimmte Zeit (bis zur Verdrängung) im Speicher (Seite nicht zweimal hintereinander)
  - abhängig von der Anzahl an verfügbaren Kacheln
- Beispiel:
  - Zugriffe: 102, 103, 601, 702, 200, 104, 602, 603, 104, 101, 701, ...
  - Seitengröße 100
  - Referenzstring (1 Kachel): 1, 6, 7, 2, 1, 6, 1, 7
  - Referenzstring (2 Kacheln): 1, 6, 7, 2, 1, 6, 7
- Referenz-Referenzstring: 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 1, 2, 0, 1, 7, 0, 1

16

Optimale Verdrängung (Belady)

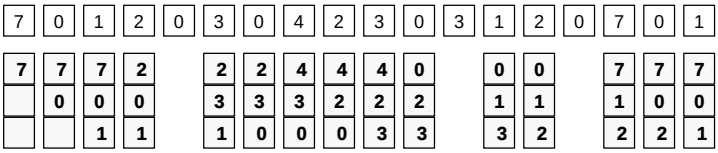
- Die Seite verdrängen, die in der Zukunft am längsten nicht gebraucht wird
  - praktisch nicht implementierbar
  - dient der Bewertung anderer Verdrängungsstrategien
- Warum ist diese Strategie optimal?
- Beispiel:



17

FIFO

- Die Seite verdrängen, die sich am längsten im Speicher befindet
- Beispiel:

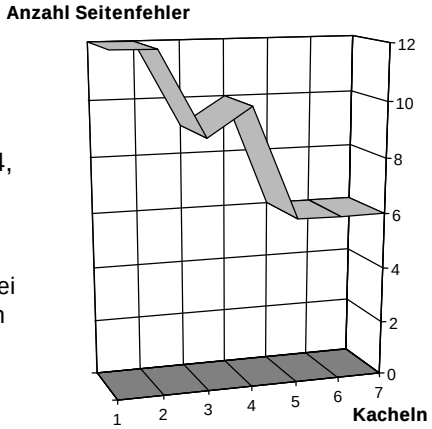


- Berücksichtigt Referenzlokalität absolut nicht
  - häufig verwendete Seite (Hot Spot) kann ausgelagert werden
  - selten referenzierte Seiten bleiben lange im Hauptspeicher
- Realisierung über Liste

18

Belady's Anomalie bei FIFO

- Referenzstring:
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- Anomalie:
  - Höhere Seitenfehlerrate bei mehr verfügbaren Kacheln



19

Least Recently Used (LRU)

- Die Seite verdrängen, die in der Vergangenheit am längsten nicht referenziert wurde
  - Referenzlokalität: Was in der Vergangenheit lange nicht referenziert wurde, wird auch in der Zukunft lange nicht referenziert
  - Projektion der Vergangenheit in die Zukunft
- LRU nähert optimale Verdrängung am besten an
- Schwierig zu implementieren

7	0	1	2	0	3	0	4	2	3	0	3	1	2	0	7	0	1
7	**7**	**7**	**2**		**2**		**4**	**4**	**4**	**0**		**0**	**2**	**2**	**2**		**1**
	0	**0**	**0**		**0**		**0**	**0**	**3**	**3**		**3**	**3**	**0**	**0**		**0**
		1	**1**		**3**		**3**	**2**	**2**	**2**		**1**	**1**	**1**	**7**		**7**

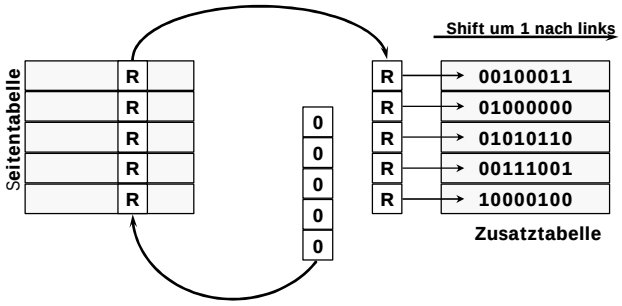
20

Implementierung von LRU

- Probleme
  - Seiten nach dem letzten Zugriff ordnen
  - **JEDE** einzelne Referenz (Lesend oder Schreibend) muß berücksichtigt werden
- Realisierung mit Hilfe eines Zählers:
  - Seitendeskriptor um ein Zählerfeld erweitern
  - CPU um eine logische Uhr erweitern (bei jedem Zugriff inkrementieren)
  - MMU überträgt bei jedem Zugriff CPU-Uhr in Seitendeskriptor (Schreiboperation)
  - Seite mit kleinstem Stempel wird verdrängt
- Realisierung mit Hilfe eines Kellers:
  - Keller referenzierter Seiten
  - Referenzierte Seite wird aus Keller entfernt und kommt an die Spitze (Keller = doppelt verkettete Liste)
  - Letzte Seite wird verdrängt

21

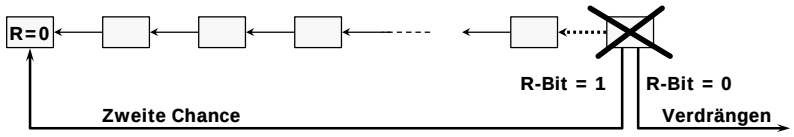
LRU-Näherung I



- Zusätzliche Tabelle:
  - Nutzung des R-Bits in der Seitentabelle
  - Periodisch (z.B. 100 msec) Übernahme der Referenzbits in die Zusatztabelle
  - Tabelleneintrag spiegelt die Referenzen in den letzten n Zyklen wider
- Seiten mit dem kleinsten Wert am längsten nicht referenziert

22

LRU-Näherung II: 2nd Chance

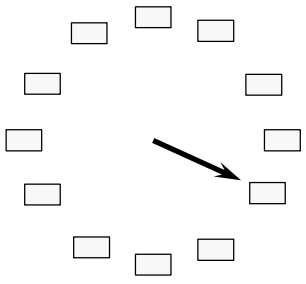


- Basiert auf FIFO-Algorithmus
- FIFO wählt Seite, die am längsten im Speicher ist
  - R-Bit = 0: Seite wird verdrängt
  - R-Bit = 1: Seite bekommt eine zweite Chance
    - Seite wird wieder an den Kopf der FIFO-Liste eingereiht
    - R-Bit wird zurückgesetzt
- 2nd Chance degeneriert zu FIFO wenn alle R-Bits gesetzt sind

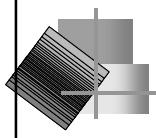
23

Clock-Algorithmus

- Implementierungsvariante des 2nd Chance
  - Aktualisierung der FIFO-Liste zu aufwendig
  - Verkettung der Seiten in einer zirkulären Liste
- "Uhrzeiger" zeigt auf die älteste Seite:
  - R-Bit = 0: Seite verdrängen
  - R-Bit = 1:
    - Zurücksetzen
    - 1 "Stunde" weiter

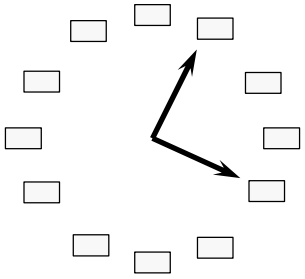


24

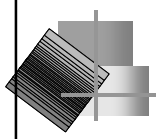


Variante des Clock-Algorithmus

- Clock-Algorithmus zu langwierig bei großem Hauptspeicher
- Zwei Zeiger:
  - Vordere Zeiger setzt R-Bit zurück
  - Hintere Zeiger prüft R-Bit
  - Beide Zeiger springen 1 Stunde vor
- Geringer Abstand zwischen Zeiger: Nur sehr häufig referenzierte Seiten bleiben
- Maximaler Zeigerabstand: Original Clock-Algorithmus
- BSD UNIX



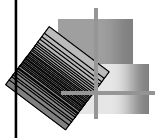
25



Weitere Verdrängungsstrategien

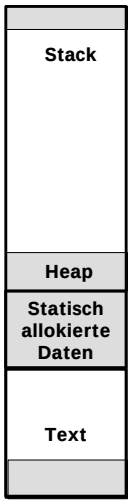
- Zufallswahl
  - Eine zufällig ausgewählte Seite wird verdrängt
- Least Frequently Use (LFU)
  - Zähler, der bei jedem Seitenzugriff inkrementiert wird
  - Seite mit dem kleinsten Zählerstand wird verdrängt
  - Problem: Kurzzeitg häufig und dann nicht mehr benötigte Seiten (z.B. Initialisierung)
  - Periodisch Links-Shift des Zählers (Exponentielle Dämpfung)
- …

26

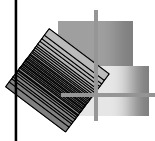


Kachelzuteilung

- Initiale Seitentabelle für neu eingerichteten Adreßraum:
  - Leer
  - Partiell besetzt
  - Vollständig initialisiert
- Demand Paging
  - Eine Seite wird erst beim ersten Zugriff (Seitenfehler) geladen
- Prepaging
  - Seiten werden vor dem ersten Zugriff geladen
  - Ausnutzung der Zugriffscharakteristik auf Platten
    - Zugriffszeit auf erste Seite lang
    - weitere Seiten (Sektoren) schnell
- Mischformen



27



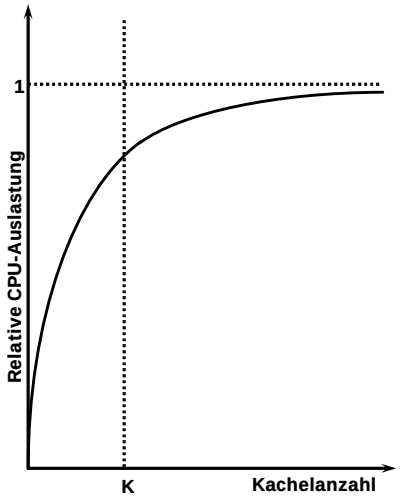
Kachelanzahl

- Wechselwirkungen:
  - Möglichst wenig Kachel zuordnen
    - mehr Adreßräume
    - freie Kacheln für Notfälle (?)
    - Nachteil: u.U. Leistungseinbußen trotz freier Kacheln
  - Möglichst viele Kacheln zuordnen
    - geringe Seitenfehlerrate
    - Nachteil: u.U. Ressourcenverschwendung
- Jedem Adreßraum gleich viele Kacheln?
- Minimum:
  - Nach Seitenfehler müssen alle Seiten geladen werden, die für die Instruktionswiederholung notwendig sind
    - 8 Seiten und mehr (z.B. Blockkopierbefehle)

28

Seitenflattern (Thrashing)

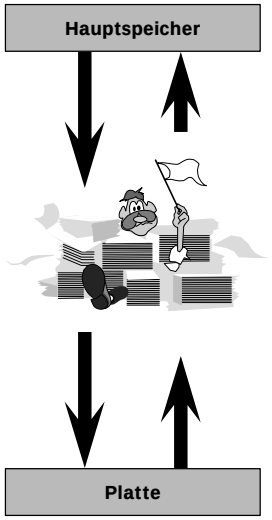
- Schwellwert K
- Mehr als K Kacheln allokiert:
  - Seitenfehlerrate klein genug
  - viele Seiten selten referenziert
  - u.U. für andere Adreßräume besser einsetzbar
- Genau K Kacheln
- Weniger als K Kacheln allokiert:
  - Kontrollfluß löst ständig Seitenfehler aus
  - Betriebssystem lädt nur Seiten nach
  - Kontrollfluß kommt praktisch nicht mehr an die Reihe
- Wie groß ist K?



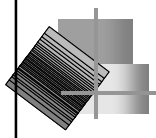
29

Seitenflattern ist ein globaler Effekt

- Seitenflattern beginnt in ein oder mehreren Adreßräumen
- Viele Ein- und Auslagerungs-aufträge für Platte
- Mittlere Zugriffszeit bei E/A wächst
- Ohne Vorkehrungen kommt praktisch das ganze System zum Stillstand



30



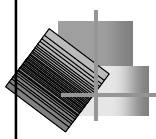
Working Sets

- Magische Schwellwert K
- P. Denning 1968
  The Working Set Model for Program Behavior
  CACM, Mai 1968
- Working-Set (WS) eines Kontrollflusses:

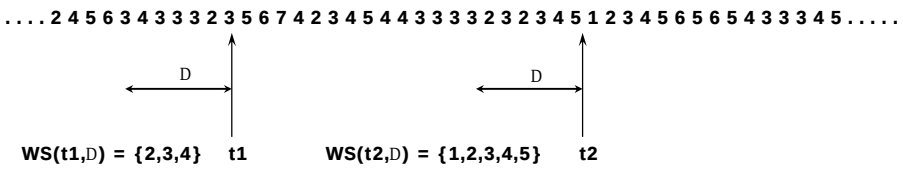
$$WS(t,\Delta) = \text{letzten } \Delta \text{ referenzierten Seiten vor } t$$

- Wahl von Δ kritisch:
  - zu klein: Lokalitätsmenge nicht enthalten
  - zu groß: mehrere Lokalitätsmengen enthalten
  - günstige Werte liegen bei ca. 10^4

31



Beispiel



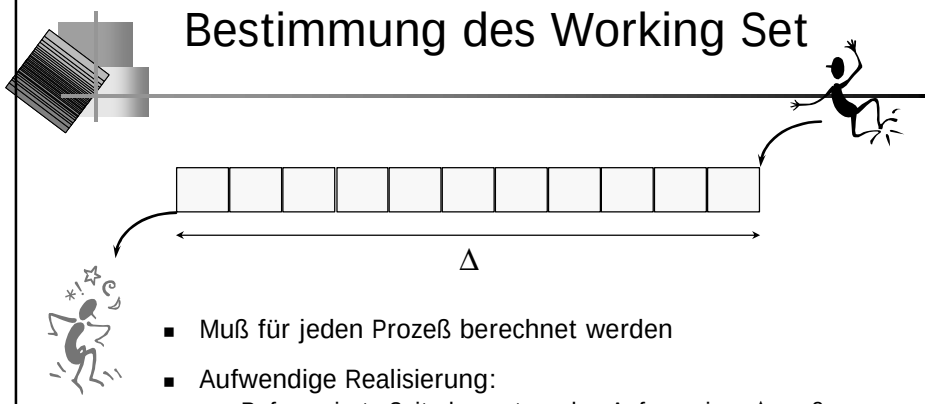
- Größe des Working-Set für Kachelzuteilung wichtig:

$$D = \sum_k |WS_k(t,\Delta)|$$

  - D > Verfügbare Kachelanzahl: Thrashing-Gefahr
    - Adreßräume auslagern (Swapping)
    - Kontrollflüsse suspendieren
  - D < Verfügbare Kachelanzahl: Weitere Kontrollflüsse aktivieren

32

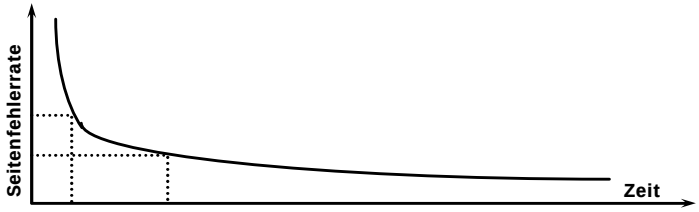
Bestimmung des Working Set



- Muß für jeden Prozeß berechnet werden
- Aufwendige Realisierung:
  - Referenzierte Seite kommt an den Anfang einer Δ großen Schlange
  - Letzte in der Schlange gespeicherte Seite fällt raus
  - Jede Seite in der Schlange ist im Working Set
- Änderungen rechtzeitig mitbekommen
- Approximation: siehe LRU-Näherung I

33

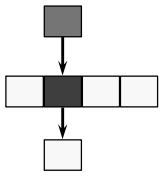
Vermeidung von Thrashing

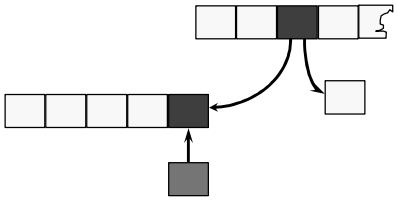


- Beobachtung der Seitenfehlerrate
- Überschreitung einer Maximalfrequenz: Thrashing
  - Zusätzliche Kacheln zuordnen
  - Adreßräume auslagern (Swapping)
- Unterschreitung einer Minimalfrequenz:
  - Kacheln entziehen

34

Globale und lokale Verdrängung





Lokale Verdrängung

- Eine Kachel des eigenen Adreßraums wird verdrängt
  - Anzahl zugewiesener Kacheln bleibt gleich
  - Kachelanzahl selbst-bestimmt
  - Working-Set sinnvoll
  - Geringerer Durchsatz (u.U. schwer, weitere Kacheln zu bekommen)

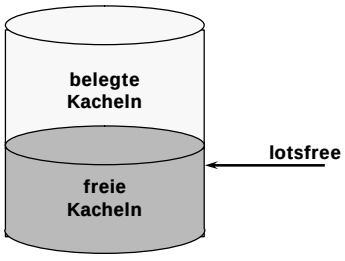
Globale Verdrängung

- Eine Kachel eines beliebigen Adreßraums wird verdrängt
  - Höherpriore Kontrollflüsse können ihren Bedarf bei anderen Adreß-räumen decken
  - Kachelanzahl hängt auch von anderen Kontrollflüssen ab
  - Höherer Durchsatz

35

UNIX: Dämon-Paging

- Seitenfehlerbehandlung und Verdrängung trennen
- Spezieller, nebenläufiger Prozeß sorgt für freie Kacheln
  - alle 250 msec aktiv
  - Schwellwert (lotsfree) an freien Kacheln unterschritten?
  - Auslagern von Seiten gemäß Clock-Algorithmus (2-Zeiger-Variante)
  - Globale Verdrängung
- lotsfree ca. 1/4 des verfügbaren Speichers



36