Betriebssysteme

Sequentielle und eng gekoppelte parallele Systeme

Kapitel 5: Speicherverwaltung

Prof. Dr. Wolfgang Küchlin

Dipl.-Inform., Dr. sc. techn. (ETH)

Arbeitsbereich Symbolisches Rechnen
Wilhelm-Schickard-Institut für Informatik
Fakultät für Informations- und Kognitionswissenschaften

Universität Tübingen

Steinbeis Transferzentrum
Objekt- und Internet-Technologien (OIT)

Wolfgang.Kuechlin@uni-tuebingen.de
http://www-sr.informatik.uni-tuebingen.de

Überblick

1. Implementierung des Seitentausches
2. Schritte des Seitentausches Seitentausch-Algorithmen
3. Das Lokalitätsprinzip
4. Optimaler Seitentausch
5. NRU – not recently used
6. FIFO
7. 2nd chance
8. Clock
9. LRU – least recently used
10. Modellierung von Seitentauschverfahren

Implementierung des Seitentausches

Anforderung an die HW

- MMU aus Effizienzgründen.
- Unterbrechbare und wieder aufsetzbare Instruktionen.
  - Bei 3-Adress Instruktion (CISC) können 3 Seitenfehler auftreten
  - z = x + y
  - Z.B. y geladen, Seitenfehler bei Laden von x
    - Instruktion wird unterbrochen
    - Seite von x wird eingelagert
    - x wird geladen
    - Instruktion wird fortgeführt

Schritte des Seitentausches

1. MMU bekommt virtuelle Adresse, die auf ungültiger Seite liegt (V = 0).
2. HW sichert Info zum Wiederaufsetzen der Instruktion. Kernel Trap.
3. Weiterer Kontextwechsel in SW (Sichern von Registern etc.); schließlich Aufruf der Seitenfehlerbehandlungsroutine PFH im BS.
4. PFH stellt die verursachende virtuelle Adresse und Seite fest (aus MMU oder indirekt aus Instruktion).

Schritte des Seitentausches

5. PFH stellt fest, ob Pseudofehler oder echter Fehler. Bei echtem Fehler SIGSEGV an Prozess und neues Scheduling; exit.

   Pseudofehler:
   - a Seite nicht eingelagert.
   - b Seite schreibgeschützt wegen copy-on-write.
   - c Seite temporär schreibgeschützt durch pager.
   - d Seite pseudo ungültig markiert,
     um R = 1 in SW zu setzen.

   c und d benötigen keinen neuen Rahmen und werden hier nicht betrachtet. Bei a und b wird ein leerer Rahmen auf der Freiliste gesucht. Ist keiner da, wird der pager gerufen, um Rahmen freizumachen.

Schritte des Seitentausches

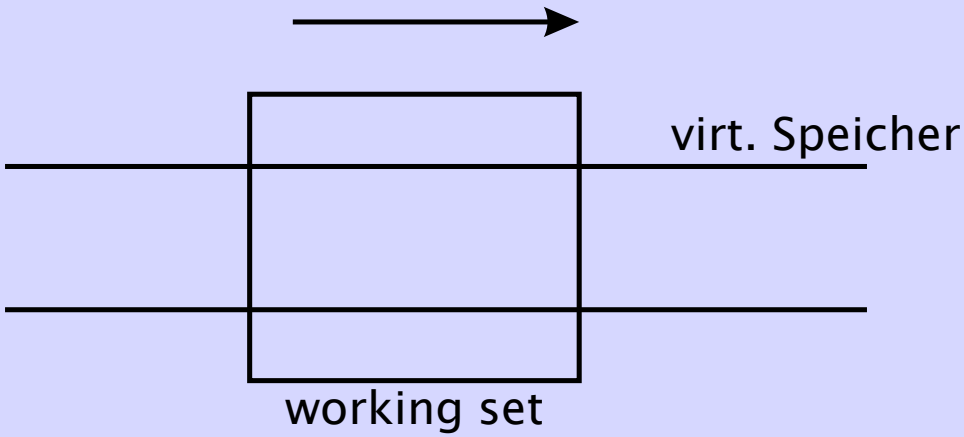
6. Ist der ausgewählte Rahmen schmutzig, oder soll der pager viele Rahmen einsammeln, so wird der verursachende Prozess suspendiert und ein anderer lauffähig gemacht. Der schmutzige Rahmen wird vom pager auf Platte zurückgeschrieben, möglichst zusammen mit anderen als Block Transfer. (Diese Rahmen werden geschützt, bis sie draußen sind.)
7. Ist ein freier Rahmen vorhanden, so findet das BS in der Tabelle virtueller Seiten des verursachenden Prozesses die Disk-Adresse der benötigten Seite und liest sie (per DMA) ein. Während der DMA kann ein anderer Prozess laufen; der Verursacher bleibt suspendiert.

11 Schritte des Seitentausches

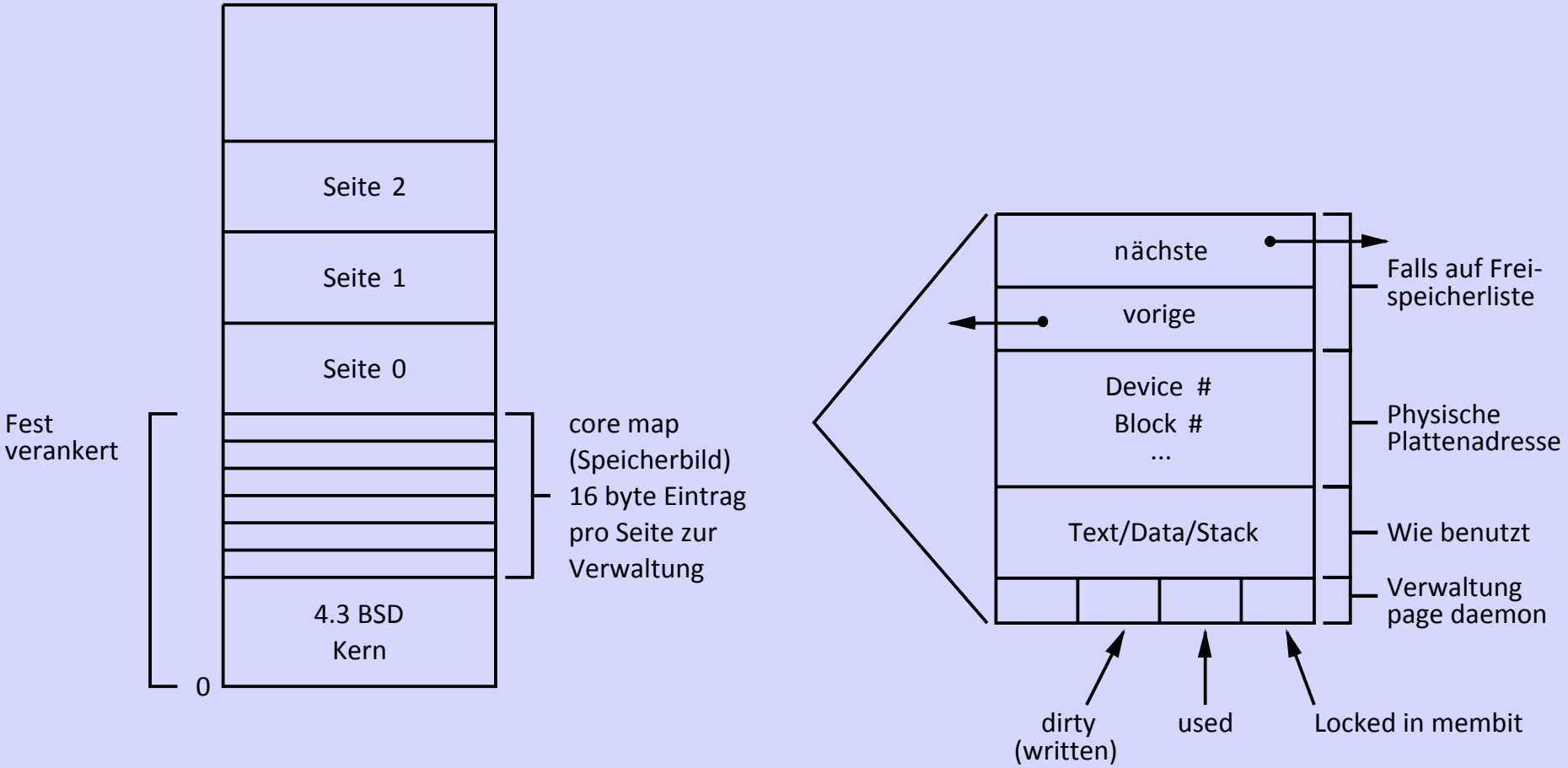
8. Ein HW interrupt signalisiert dem BS, dass die Seite da ist. Die Seitentabelle des Verursachers wird nachgeführt.
9. Der Verursacher wird in den Stand vor dem Fehler gebracht (Rücksetzen PC, Laden seiner Register).
10. Der Verursacher wird lauffähig gemacht und fährt mit der Ausführung fort.

Das Lokalitätsprinzip

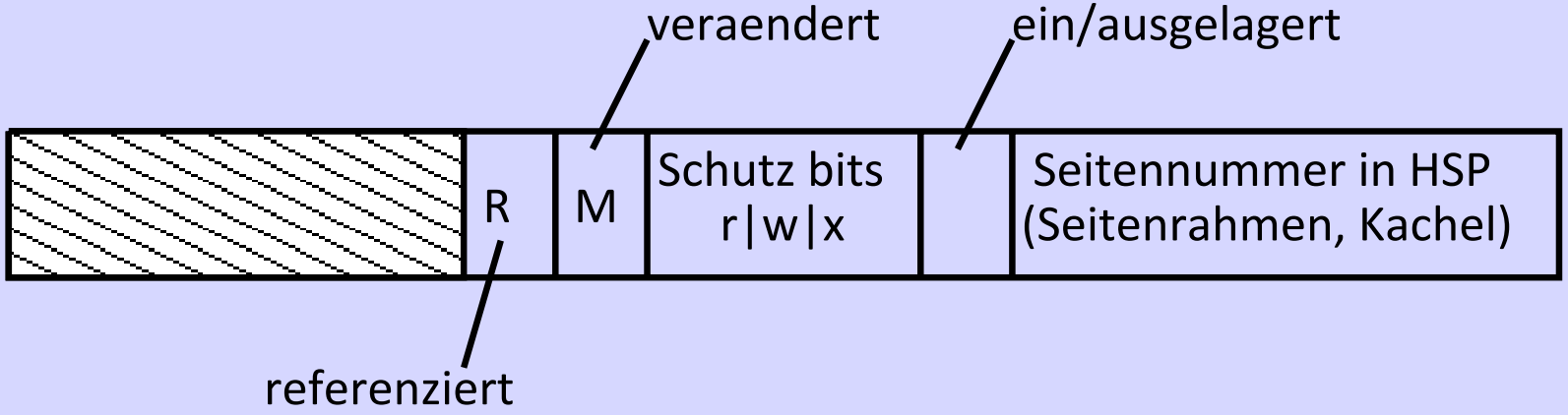
- *working set* (Arbeitsmenge): die momentan gebrauchten Seiten.
- Lokalität: Programme bleiben lange bei einer Arbeitsmenge.
- Seitenfehler treten auf beim Wechsel der Arbeitsmenge.
- *threshing* („dreschen“): Exzessives Seitentauschen, wenn der HSP zu klein → Prozess auslagern (swappen)
- Prozess muss am Anfang seine Arbeitsmenge aufbauen:
  - *demand paging* – über Seitenfehler
  - *prepaging* – Blocktransfer



Seiteneinteilung im Hauptspeicher (UNIX)



Optimaler Seitentausch



- Diejenige Seite wird ersetzt, die am spätesten referenziert werden wird.
  → Maximierung des Intervalls ohne Seitentausch.
- Nicht konstruktiv möglich, außer z.B. für externe Pager.
- Approximation durch Beobachtung der Vergangenheit.

NRU – not recently used

- Periodisch wird das R Bit zurückgesetzt
  → Finden kürzlich veränderter Seiten.
- RM Bits geben Seiten eine Priorität 0-3.
- NRU tauscht „unwichtige“ Seite gegen die benötigte.

FIFO

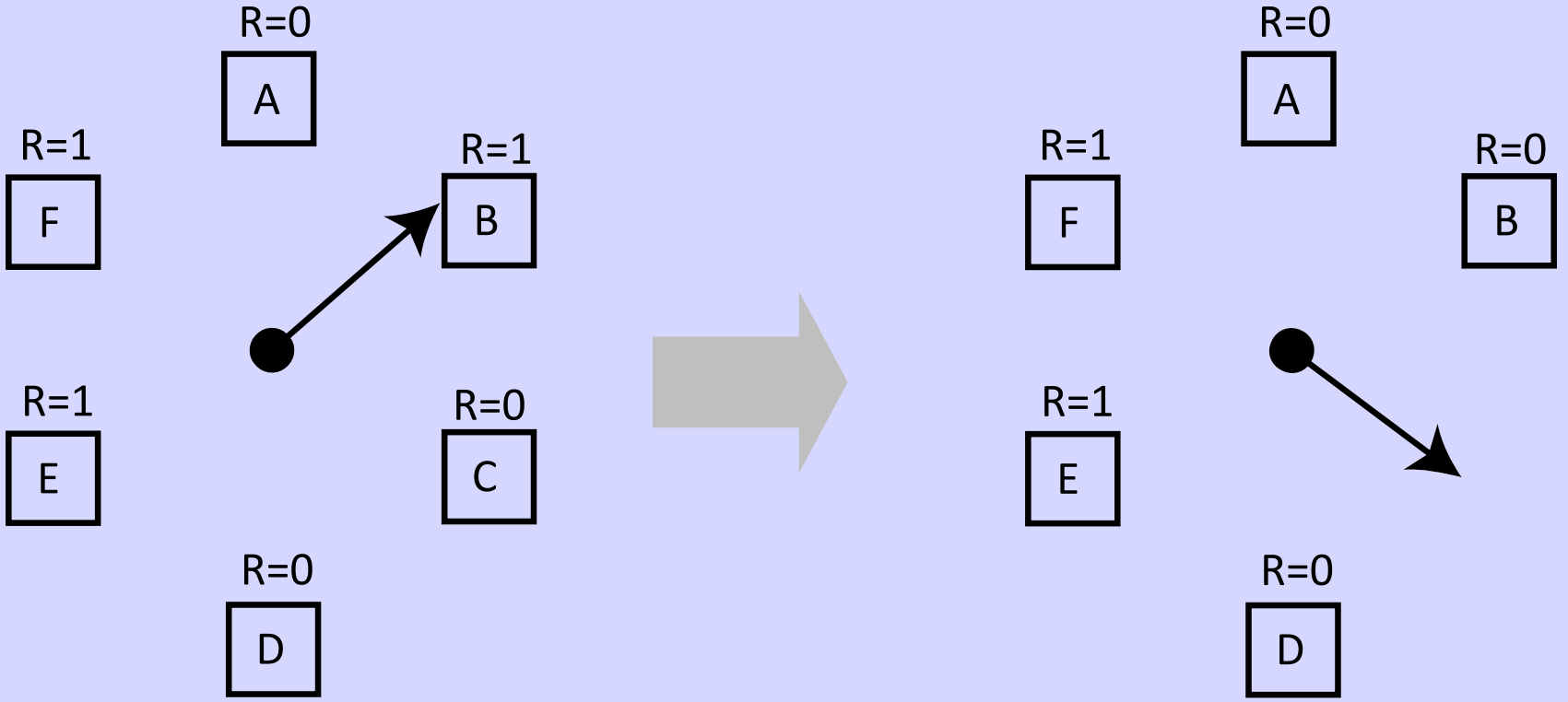
- Die am längsten im Speicher befindliche Seite wird ersetzt.
- Geringer Aufwand (+)
- Alte Seiten können auch wichtig sein (-)

2nd chance

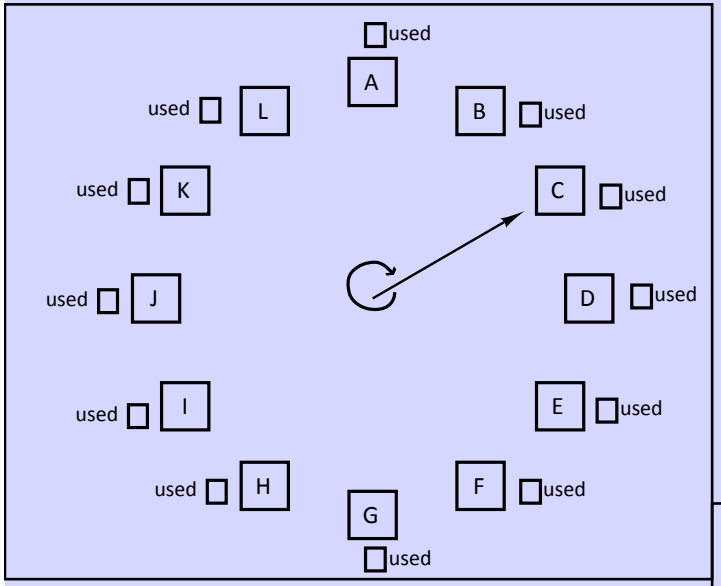
- Modifikation von FIFO
- Sortieren der Seiten nach dem Alter.
- Auswahl Löschkandidat nach FIFO
  - Falls R==0: Seite wird gelöscht
  - Falls R==1: Zweite Chance
    → setze R = 0 und Ladezeit = momentane Zeit
- → Es wird die älteste Seite mit R==0 gelöscht
- → Sonst wird die älteste Seite gelöscht.

6 Clock

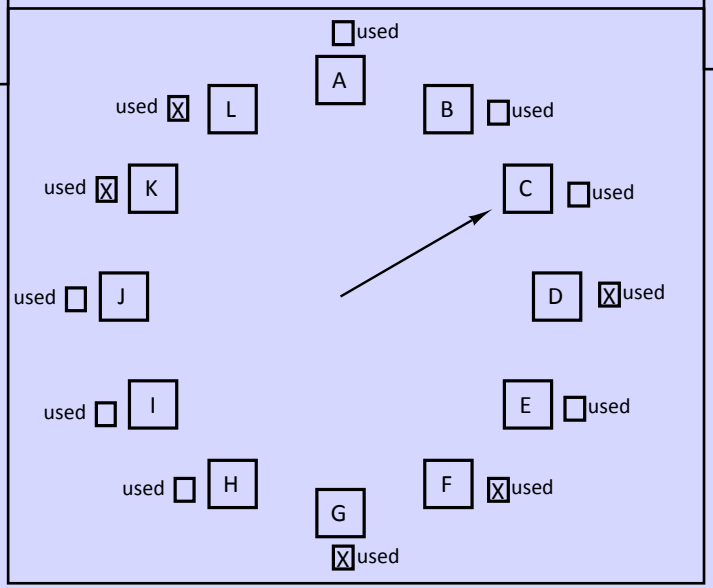
- Zeiger rotiert über die Seiten im Hauptspeicher.
- Seite wird ausgelagert, falls R==0; sonst setzt man R=0.

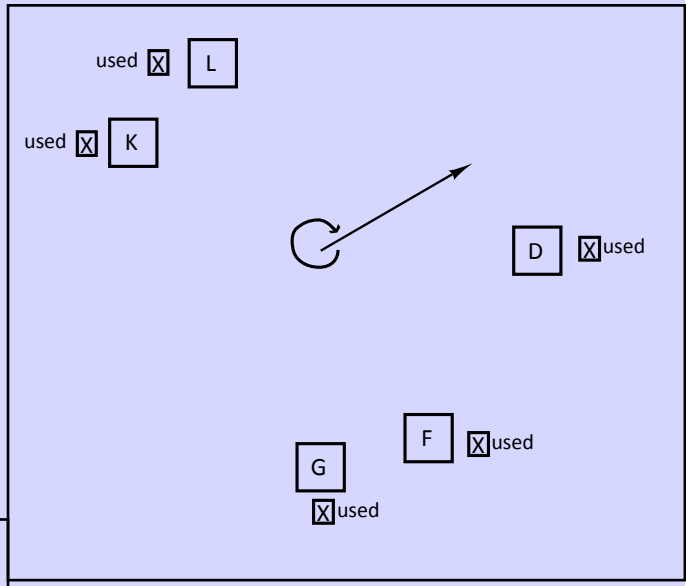


Seitentausch-Algorithmus



1. Durchlauf:
Alle „benutzt“ Bits werden zurückgesetzt





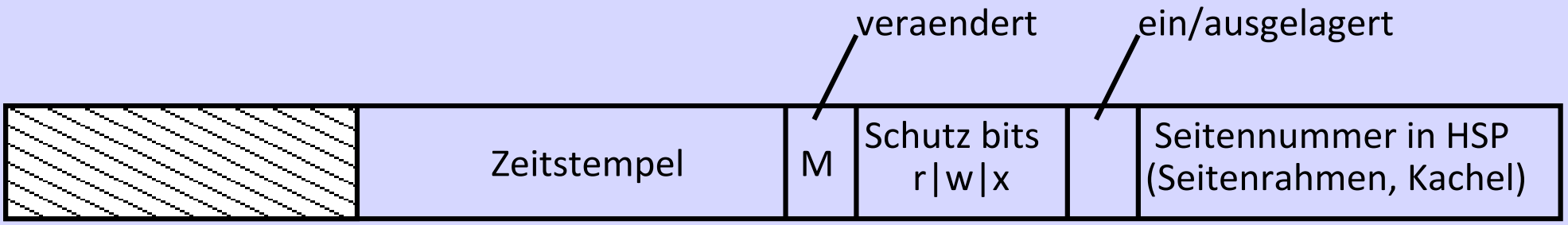
2. Durchlauf:
Alle unbenutzt Seiten werden freigesetzt.

6 Zwei Zeiger Clock (UNIX)

- Bei großem Speicher dauert ein Zeigerumlauf sehr lange.
- Daher viele Seiten referenziert, wenige ausgelagert→
- → Es werden 2 Zeiger benutzt
  - Erster Zeiger setzt R=0.
  - Zweiter Zeiger folgt erstem in kurzem Abstand, lagert Seite aus, falls R==0.

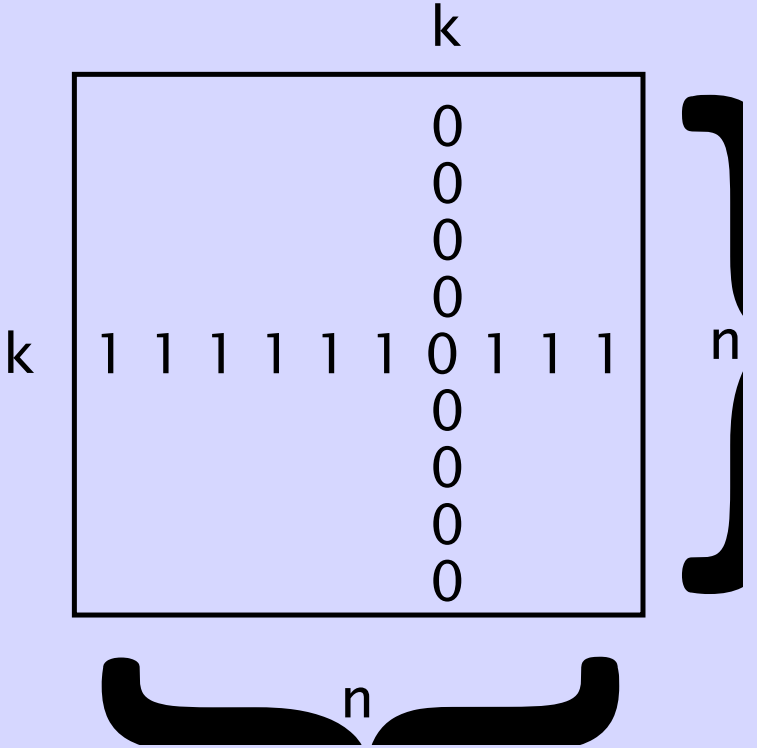
LRU – least recently used

- Tauscht diejenige Seite, die am längsten nicht benutzt wurde.
- Approximation in Hardware - Alternative 1
  - Statt R Bit Zeitstempel im Seitenrahmen speichern
  - z.B. 64 Bit Zyklenzähler
  - Rahmen mit kleinstem Zähler wird getauscht
  - Durchsuchen aller Rahmen ist teuer. (-)



LRU – least recently used

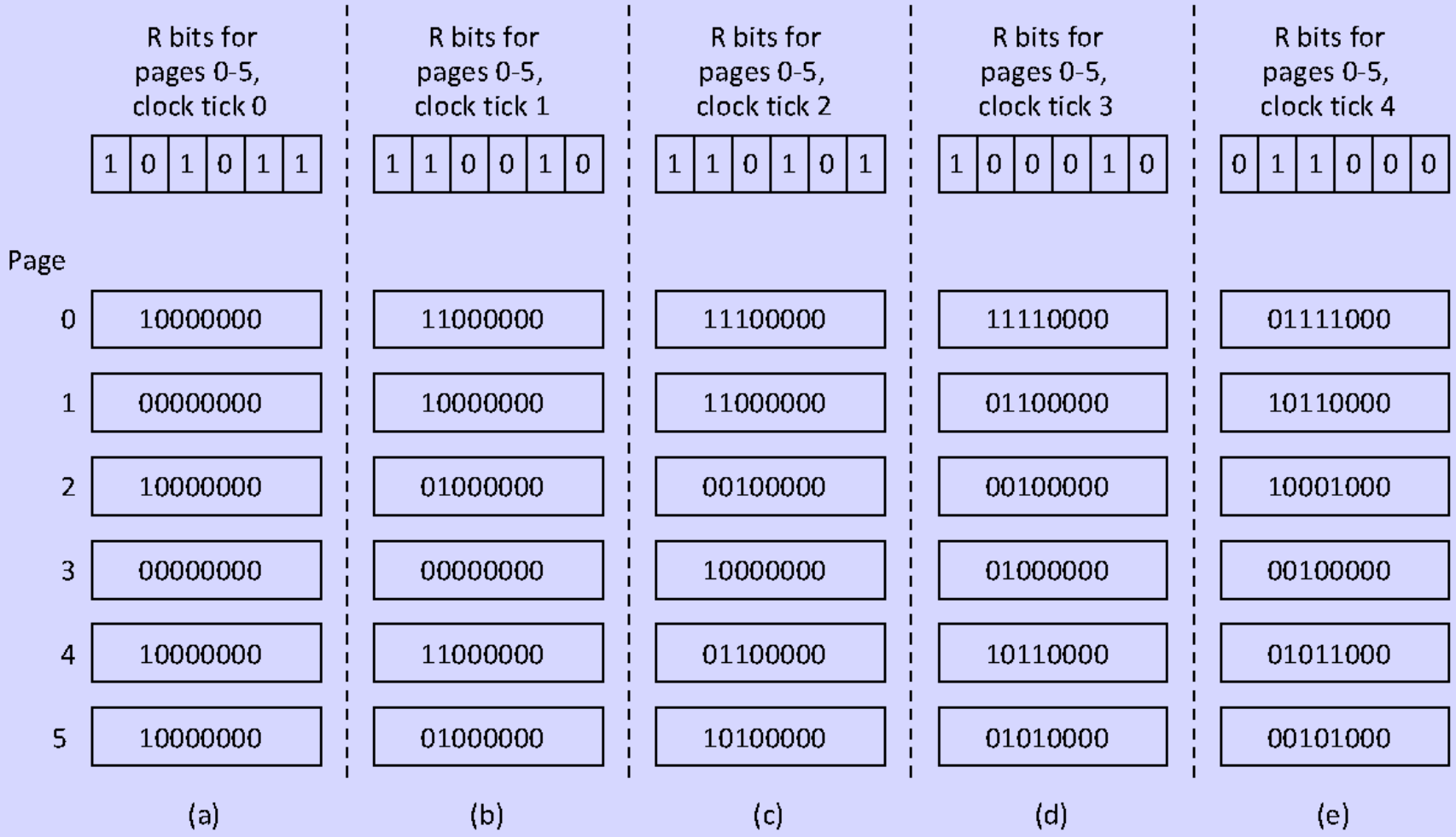
- Approximation in Hardware - Alternative 2
  - Bei *n* Rahmen Matrix mit *n*×*n* Bits.
  - Wird Rahmen *k* referenziert:
    - HW setzt Zeile *k* auf 111…1
    - HW setzt Spalte *k* auf 00…0
  - Kleinste Zeile entspricht ältestem Rahmen.
  - Probleme:
    - Größe des Arrays
      128MB HSP = 128K Rahmen
      → $2^{17} \times 2^{17}$ Bit = 16Gb = 2GB
    - Kosten der HW
    - Portabilität



LRU – least recently used

- Approximation in Software - Alternative 1
  - Rahmen werden zu Liste verkettet
  - Referenzierte Rahmen kommen nach vorne
  - Ältester Rahmen steht hinten
  - Problem: Umketten bei fast jeder Referenz → zu teuer
- Approximation in Software - Alternative 2
  - Eingreifen in SW nur noch nach jeder Zeitscheibe und HW setzt das R-Bit
  - *Aging*: Speichern eines Geschichtsvektors **v**, der die R-Bits der letzten k Zeitscheiben speichert.
    - Berechne $V=(R_i << (k-1)) \mid (V >> 1)$ in jeder Zeitscheibe.
    - LRU Seite hat kleinsten Vektor
  - Problem: Hoher Aufwand bei großem Speicher

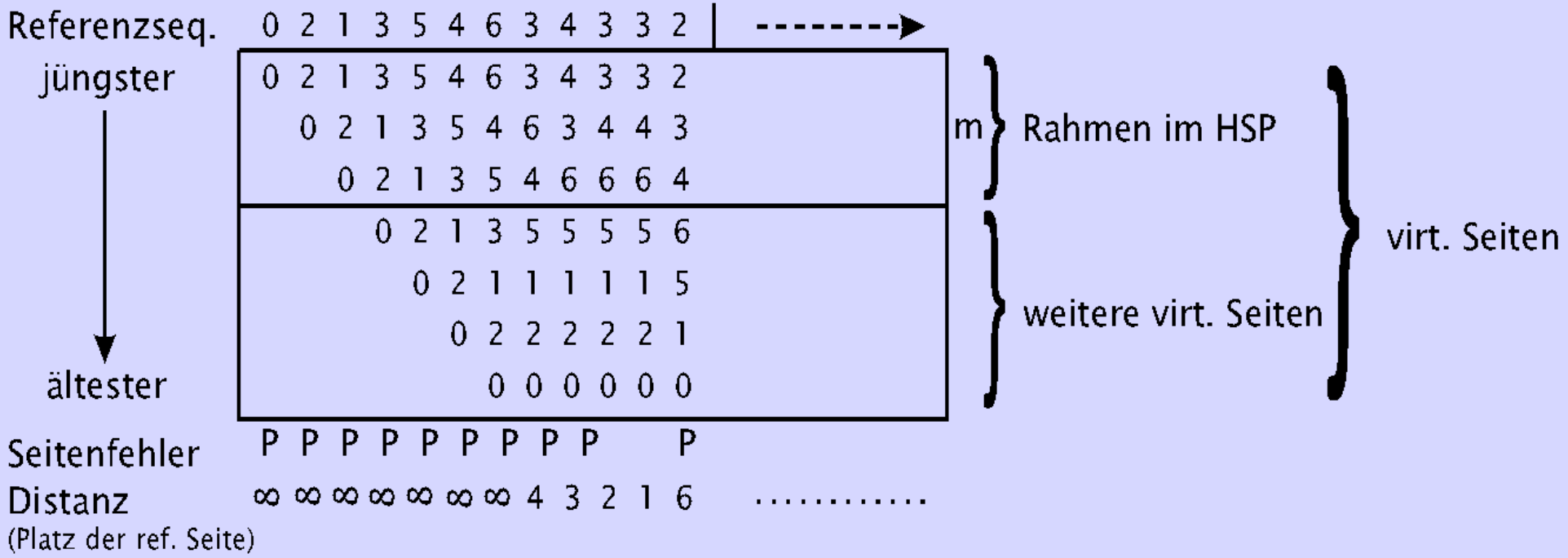
LRU – least recently used: aging



Tanenbaum, Abbildung 4-19: Der aging Algorithmus

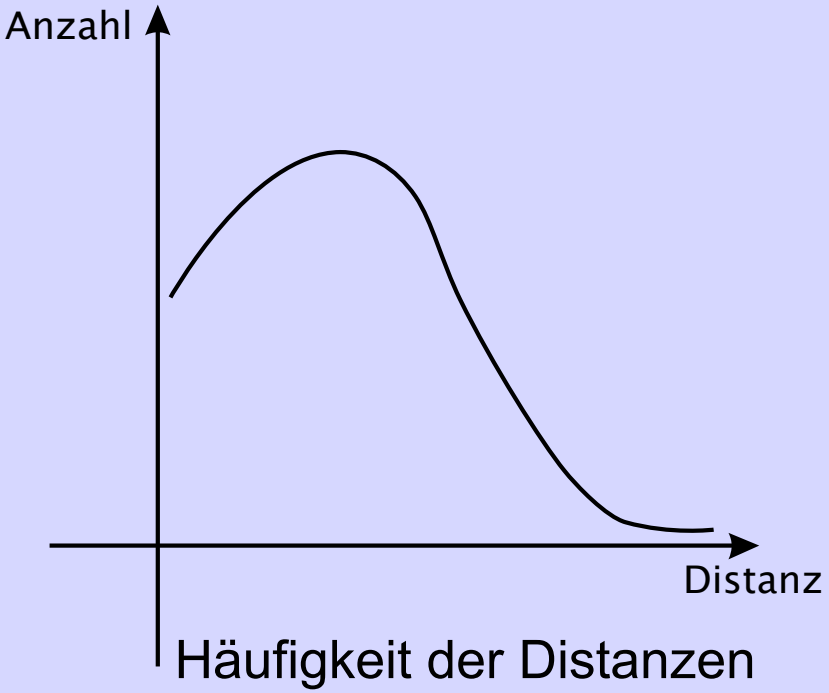
Modellierung von Seitentauschverfahren

Modellierung der Seitentausch-Aktivität des LRU-Verfahrens
bei gegebener Folge von Seiten-Referenzierungen und 3 HSP-Rahmen

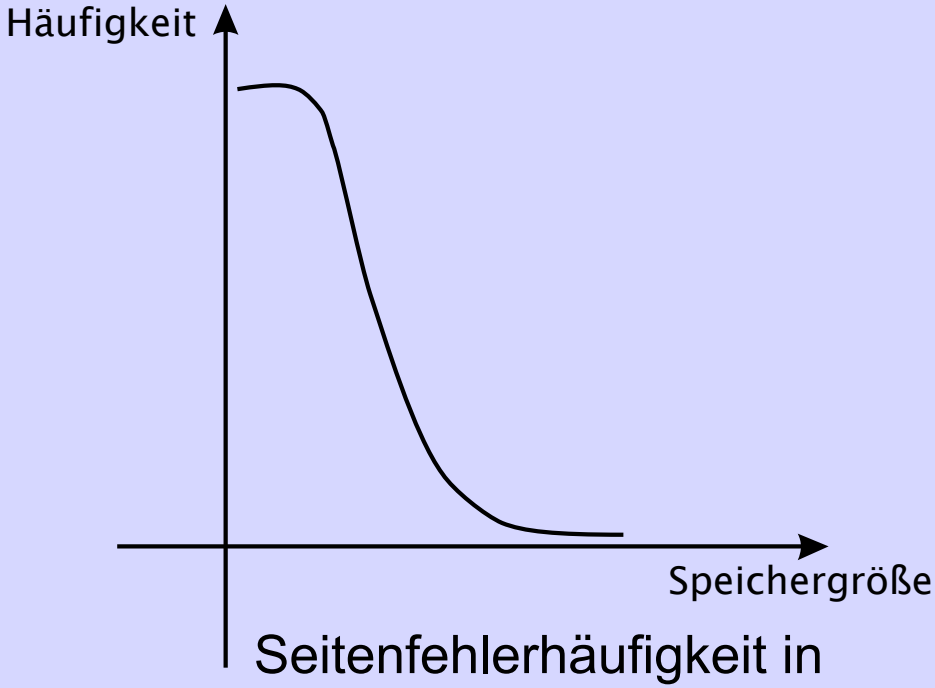


Durch Simulation mit verschiedenen m kann jeweils Anzahl der Seitenfehler bestimmt werden. Distanz <m bedeutet: kein Seitenfehler bei m Rahmen.

Das Lokalitätsprinzip



Häufigkeit der Distanzen



Seitenfehlerhäufigkeit in Abhängigkeit von der Speichergröße.

Gesucht: Optimum

Vermeiden von Seitenfehlern und gleichzeitig geringer Verbrauch von Seiten