Verdrängung eingelagerter Fragmente

Platz schaffen zur Einlagerung anderer Fragmente (d.h., Seiten oder Segmente)

Konsequenz zur Durchsetzung der Ladestregie bei Arbeitsspeichermangel

- wenn eine Überbelegung des Arbeitsspeichers vorliegt
  - der Speicherbedarf aller Prozesse ist größer als der verfügbare RAM
- aber auch im Falle (extensiver) externer Fragmentierung

OPT (optimales Verfahren) Verdrängung der Seite, die am längsten nicht mehr verwendet werden wird — unrealistisch

- erfordert Wissen über die im weiteren Verlauf der Ausführung von Prozessen generierten Speicheradressen, was bedeutet:
  - das Laufzeitverhalten von Prozessen ist im Voraus bekannt
  - Eingabewerte sind vorherbestimmt
  - asynchrone Programmunterbrechungen sind vorhersagbar
- bestenfalls ist eine gute Approximation möglich/umsetzbar

☞ Seitenfehlerwahrscheinlichkeit senken und Seitenfehlerrate verringern

Ersetzungsverfahren

Praxistaugliche Herangehensweisen

Approximation des optimalen Verfahren greift auf Wissen aus der Vergangenheit/Gegenwart zurück, d.h., ersetzt wird . . .

FIFO (*first-in, first-out*) das zuerst eingelagerte Fragment

- Fragmente entsprechend des Einlagerungszeitpunkts verketten

LFU (*least frequently used*) das am seltenste genutzte Fragment

- jeden Zugriff auf eingelagerte Fragmente zählen
- Alternative: MFU (*most frequently used*)

LRU (*least recently used*) das kürzlich am wenigsten genutzte Fragment

- Zeitstempel, Stapeltechniken oder Referenzlisten einsetzen
- bzw. weniger aufwändig durch Referenzbits approximieren

☞ die zu ersetzenden/verdrängenden Fragmente sind vorzugsweise Seiten

Zählverfahren

Auswahlkriterium ist die Häufigkeit von Seitenreferenzen

Zähler basierte Ansätze führen Buch darüber, wie häufig eine Seite innerhalb einer bestimmten Zeitspanne referenziert worden ist:

- im Seitendeskriptor ist dazu ein Referenzzähler enthalten
- der Zähler wird mit jeder Referenz zu der Seite inkrementiert
- aufwändige Implementierung, bei mäßiger Approximation von OPT

LFU ersetzt die Seite mit dem kleinsten Zählerwert

- Annahme: aktive Seiten haben große Zählerwerte und weniger aktive bzw. inaktive Seiten haben kleine Zählerwerte
- große Zählerwerte können dann aber auch jene Seiten haben, die z.B. nur in der Initialisierungsphase aktiv gewesen sind

MFU ersetzt die Seite mit dem größten Zählerwert

- Annahme: kürzlich aktive Seiten haben eher kleine Zählerwerte

Zeitverfahren

Auswahlkriterium ist die Zeitspanne zurückliegenden Seitenreferenzen

LRU_{time} verwendet einen Zähler („logische Uhr“) in der CPU, der bei jedem Speicherzugriff erhöht und in den zugehörigen Seitendeskriptor geschrieben wird

- verdrängt die Seite mit dem kleinsten Zeitstempelwert

LRU_{stack} nutzt einen Stapel eingelagerter Seiten, aus dem bei jedem Seitenzugriff die betreffende Seite herausgezogen und wieder oben drauf gelegt wird

- verdrängt die Seite am Stapelboden

LRU_{ref} führt Buch über alle zurückliegenden Seitenreferenzen

- verdrängt die Seite mit dem größten Rückwärtsabstand

- entspricht OPT, wenn allerdings die Vergangenheit betrachtet wird
  - gute Approximation von OPT, bei sehr aufwändiger Implementierung

Approximation von LRU

Alterung von Seiten (engl. *page aging*) verfolgen

Referenzbit (engl. *reference bit*) im Seitendeskriptor zeigt an, ob auf die zugehörige Seite zugegriffen wurde:

0 $\mapsto$ kein Zugriff
1 $\mapsto$ Zugriff bzw. Einlagerung

- das Alter eingelagerter Seiten wird periodisch (Zeitgeber) bestimmt:
  - für jede eingelagerte Seite wird ein N-Bit Zähler (*age counter*) geführt
  - der Zähler ist als Schieberegister (engl. *shift register*) implementiert
  - nach Aufnahme eines Referenzbits in den Zähler, wird es gelöscht
- „kürzlich am wenigsten genutzte" Seiten haben den Zählerwert 0
  - d.h., sie wurden seit N Perioden nicht mehr referenziert ($\rightarrow$ NT)

Approximation von LRU (Forts.)

Schieberegistertechnik zur Bestimmung des Lebensalters einer Seite

page aging (Forts.) mit Ablauf eines Zeitquantums (Tick), werden die Referenzbits eingelagerter Seiten des laufenden Prozesses in die Schieberegister seiner Seitendeskriptoren übernommen

- z.B. ein 8-Bit „*age counter*": $age = (age >> 1) \,|\, (ref << 7)$

Referenzbit	Alter (*age*, initial 0)
1	10000000
1	11000000
0	01100000
1	10110000
⋮	⋮

Den Inhalt des 8-Bit Schieberegisters (*age*) als ganze Zahl interpretiert liefert ein Maß für die Aktivität einer Seite: mit abnehmendem Betrag (sinkender Aktivität) steigt die Ersetzungspriorität.

☞ Aufwand steigt mit der Adressraumgröße des unterbrochenen Prozesses

Approximation von LRU (Forts.)

Referenzierte Seiten sind vermeintlich aktive Seiten

second chance (auch: *clock policy*)

- arbeitet im Grunde nach FIFO, berücksichtigt jedoch zusätzlich noch die Referenzbits der jeweils in Betracht zu ziehenden Seiten
- periodisch (Zeitgeber, Tick) werden die Seitendeskriptoren des (unterbrochenen) laufenden Prozesses untersucht

Referenzbit	Aktion	Bedeutung
1	Referenzbit auf 0 setzen	Seite erhält zweite Chance
0	—	Seite ist Ersetzungskandidat

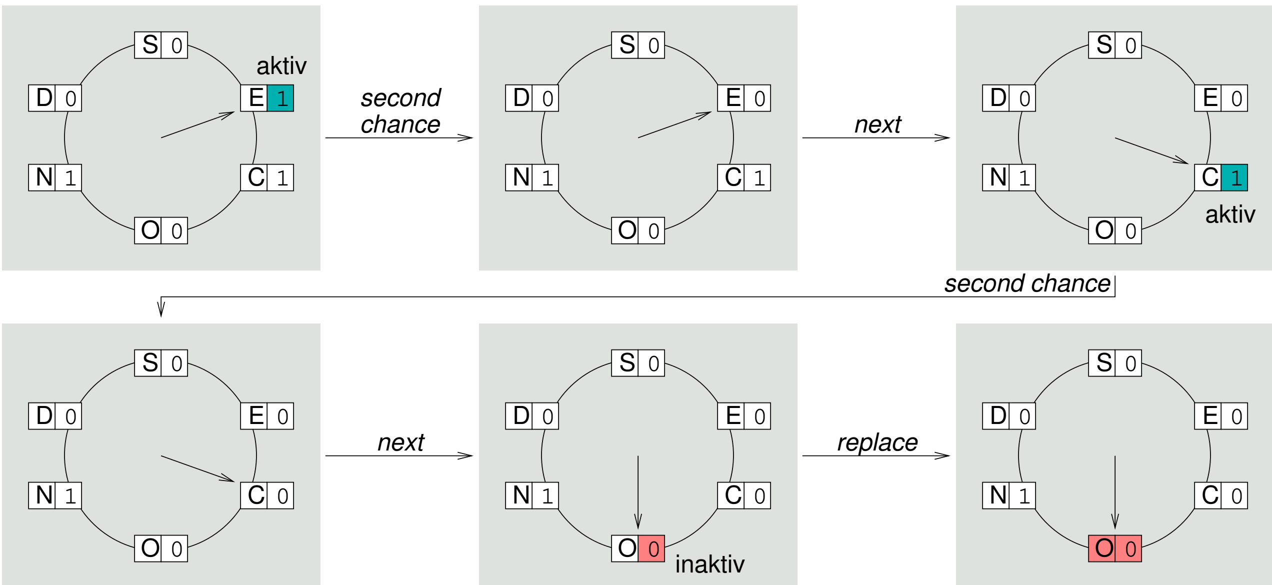
- im schlimmsten Fall erfolgt ein Rundumschlag über alle Seiten, falls die Referenzbits aller betrachteten Seiten (auf 1) gesetzt waren
  - die Strategie „entartet" dann zu FIFO

☞ unterscheidet nicht zwischen lesende und schreibende Seitenzugriffe

12 Arbeitsspeicher 12.4 Ersetzungsstrategie

Approximation von LRU (Forts.)

Clock Policy



Approximation von LRU (Forts.)

Schreibzugriffe stärker gewichten als Lesezugriffe

enhanced second chance prüft zusätzlich, ob eine Seite schreibend oder nur lesend referenziert wurde

- Grundlage dafür ist ein Modifikationsbit (*modify/dirty bit*)
  - ist als weiteres Attribut in jedem Seitendeskriptor enthalten
  - wird bei Schreibzugriffen auf 1 gesetzt, bleibt sonst unverändert
- zusammen mit dem Referenzbit ergeben sich vier Paarungen (R, D):

	Bedeutung	Entscheidung
(0, 0)	ungenutzt	beste Wahl
(0, 1)	beschrieben	keine schlechte Wahl
(1, 0)	kürzlich gelesen	keine gute Wahl
(1, 1)	kürzlich beschrieben	schlechteste Wahl

- kann für jeden Prozess(adressraum) zwei Umläufe erwirken
  - gibt referenzierten Seiten damit eine dritte Chance (→ MacOS)

Freiseitenpuffer

Reserve freier (d.h. ungebundener) Seitenrahmen garantieren

Alternative zur Seitenersetzung, die den Vorabruf von Seiten nutzt

- Steuerung der Seitenüberlagerung über Schwellwerte („*water mark*“):
  - *low* Seitenrahmen als frei markieren, Seiten ggf. auslagern
  - *high* Seiten ggf. einlagern, Vorausladen
- die Auswahl greift z.B. auf Referenz-/Modifikationsbits zurück

Freiseiten gelangen in einen Zwischenspeicher (engl. *cache*)

- die Zuordnung von Seiten zu Seitenrahmen bleibt jedoch erhalten
- vor ihrer Ersetzung doch noch benutzte Seiten werden „zurückgeholt“
- sog. „*reclaiming*“ von Seiten durch Prozesse (→ Solaris, Linux)

☞ vergleichsweise effizient: die Ersetzungszeit entspricht der Ladezeit

Seitenanforderung

Verteilung von Seitenrahmen auf Prozessadressräume

Prozessen ist mindestens die kritische Masse von Seitenrahmen zur Verfügung zu stellen, um in ihrer Ausführung voranschreiten zu können

- Rechnerausstattung/-architektur geben „harte" Begrenzungen vor:
  - **Obergrenze** die Größe des Arbeitsspeichers
  - **Untergrenze** definiert durch den komplexesten Maschinenbefehl
    - schlimmster Fall möglicher Seitenfehler (S. IX-19)
- innerhalb dieser Grenzen, ist die Zuordnung ...
  - **gleichverteilt** in Abhängigkeit von der Prozessanzahl und/oder
  - **größenabhängig** bedingt durch den (statischen) Programmumfang
- „weiche" Begrenzungen ergeben sich z.B. durch die gegenwärtige Systemlast und den gewünschten Grad an Mehrprogrammbetrieb

Einzugsbereiche

Wirkungskreis der Seitenüberlagerung

lokal nur Seitenrahmen des von der Seitenersetzung betroffenen Prozessadressraums nutzen ($\rightarrow$ NT)

- die Seitenfehlerrate ist von einem Prozess selbst kontrollierbar
  - Prozesse verdrängen niemals Seiten anderer Adressräume
  - fördert ein deterministisches Laufzeitverhalten von Prozessen
- statische Zuordnung von Seitenrahmen zum Prozessadressraum

global alle verfügbaren Seitenrahmen heranziehen; dynamische Zuordnung

- Verdrängung/Ersetzung von Seitenrahmen ist unvorhersehbar und auch nicht bzw. nur schwer reproduzierbar
  - Seitenfehler erhalten *Interrupt*-Eigenschaften
- der zeitliche Ablauf einer Programmausführung ist abhängig von den in anderen Adressräumen vorgehenden Aktivitäten

- Kombination beider Ansätze ist möglich: Prozess-/Adressraumklassen
  - z.B. nur Echtzeitprozesse der lokalen Seitenersetzung unterziehen

Seitenflattern
(engl. *thrashing*)

„Dresche beziehen“ — wenn durch Seitenüberlagerung verursachte E/A die gesamten Systemaktivitäten bestimmt

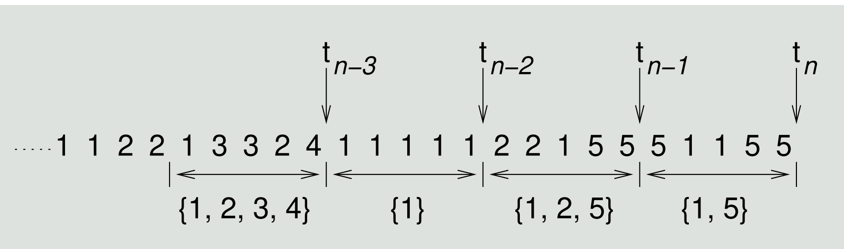
- eben erst ausgelagerte Seiten werden sofort wieder eingelagert
  - Prozesse verbringen mehr Zeit beim „*paging*“ als beim Rechnen
  - das Problem ist immer in Relation zu der Zeit zu setzen, die Prozesse mit sinnvoller Arbeit verbringen
- ein mögliches Phänomen der globalen Seitenersetzung
  - Prozesse bewegen sich zu nahe am Seiten(rahmen)minimum
  - zu hoher Grad an Mehrprogrammbetrieb
  - ungünstige Ersetzungsstrategie
- verschwindet ggf. so plötzlich von allein, wie es aufgetreten ist ...

☞ Umlagerung (*Swapping*) von Adressräumen bzw. Arbeitsmengen

Seitenflattern vermeiden
Heuristik über zukünftig erwartete Seitenzugriffe erstellen

Arbeitsmenge (engl. *working set*) die Menge der Seiten, die ein Prozess lokal/das System global in naher Zukunft aktiv in Benutzung haben wird

- Berechnung der Arbeitsmenge ist nur näherungsweise möglich:
  - Ausgangspunkt ist die Seitenreferenzfolge der jüngeren Vergangenheit
  - regelmäßig wird ein Fenster (*working set window*) darauf geöffnet



- zu kleine Fenster haben wenig aktive Seiten
- zu große Fenster zeigen Überlappungen

- die Fensterbreite gibt eine „feste Anzahl von Maschinenbefehlen“
  - sie wird approximiert durch periodische Unterbrechungen
  - die Befehlsanzahl ergibt sich in etwa aus der Periodenlänge

Seitenflattern vermeiden (Forts.)

Approximation der Arbeitsmenge

Grundlage sind Referenzbit, Seitenalter und periodische Unterbrechungen

- bei jedem Tick werden die Referenzbits eingelagerter Seiten geprüft:
  1 $\rightsquigarrow$ Referenzbit und Alter auf 0 setzen
  0 $\rightsquigarrow$ Alter der Seite um 1 erhöhen
  - Alterswerte von Arbeitsmengenseiten sind kleiner als die Fensterbreite
- nur Seiten des laufenden Prozesses altern $\models$ lokale Arbeitsmenge
  - Problem: gemeinsam genutzte Seiten (z.B. *shared libraries*)
- Seiten aller „aktiven Adressräume" altern $\models$ globale Arbeitsmenge
  - Problem: vergleichsweise (sehr) hoher Systemaufwand

Umlagerung bzw. Vorausladen kompletter Arbeitsmengen praktizieren:

- Mindestmenge von Seitenrahmen laufbereiter Prozesse vorhalten
- Prozesse mit unvollständigen Arbeitsmengen stoppen/suspendieren

Speicherverwaltung

Buchführung über freie/belegte Arbeitsspeicherfragmente

- Programme erhalten Arbeitsspeicher statisch/dynamisch zugeteilt
  - Zuteilungseinheiten sind Segmente oder Seitenrahmen
- Zuteilung von Arbeitsspeicher ist Aufgabe der Platzierungsstrategie
  - die Erfassung freier Fragmente hängt u.a. ab vom Adressraummodell
    - Seitenrahmen $\rightsquigarrow$ Bitkarte, Segmente $\rightsquigarrow$ Löcherliste
  - Zuteilungsverfahren: *best*/*worst*-fit, *buddy*, *first*/*next*-fit
  - Verschmelzung/Kompaktifizierung wirkt ext. Fragmentierung entgegen
- die Ladestrategie sorgt für die Einlagerung von Seiten (Segmenten)
  - auf Anforderung oder im Voraus
- eingelagerte Seiten unterliegen der Ersetzungsstrategie
  - Ersetzungsverfahren: OPT, FIFO, LFU, MFU, LRU (*clock*)
    - alternativer Ansatz ist der Freiseitenpuffer
  - Verdrängung arbeitet adressraumlokal oder systemglobal
  - Arbeitsmengen auseinanderreißen kann zum Seitenflattern führen