LUENTO 9

MUISTINHALLINNAN OHJELMISTO: Sivutuksen algoritmit

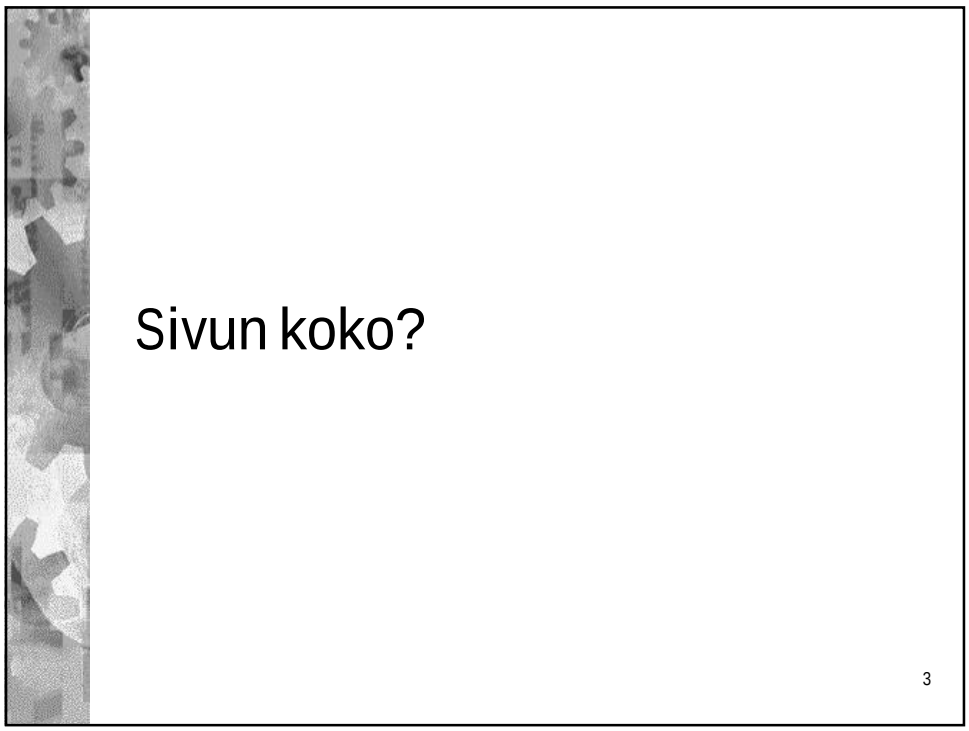
Stallings, Luku 8.2

1

Perusvalinnat

- Virtuaalimuistia vai ei?
  - originaali UNIX ja MS-DOS eivät käyttäneet
    - ei tarvittavaa laitteistotukea (MMU, TLB)
- Sivutus vai segmentointi?
  - pelkkään segmentointiin perustuvat häviämässä
  - molempien yhteiskäyttö yleistymässä
  - nykyisin lähes aina monta tasoa
- Millaisia algoritmeja käytetään?
  - monta tekijää vaikuttamassa
    - reaaliaikaisuus? palvelinkone?
    - laitteistotuki?
- Jatkossa käsitellään vain sivutusta

2


Sivun koko?
3

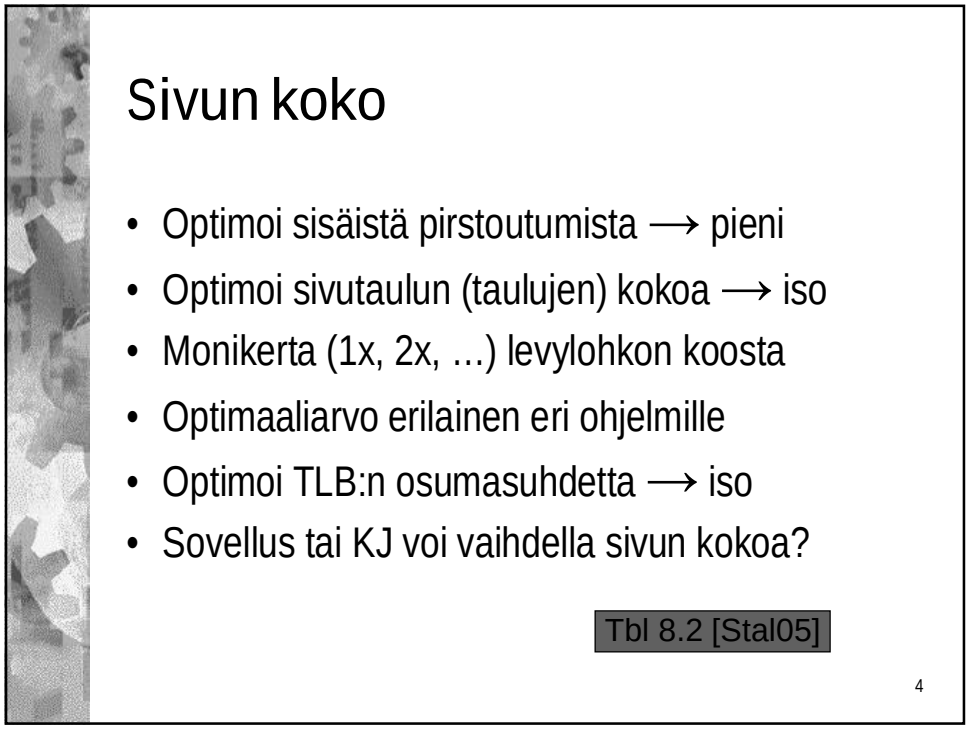

Sivun koko
• Optimoi sisäistä pirstoutumista ⟶ pieni
• Optimoi sivutaulun (taulujen) kokoa ⟶ iso
• Monikerta (1x, 2x, …) levylohkon koosta
• Optimaaliarvo erilainen eri ohjelmille
• Optimoi TLB:n osumasuhdetta ⟶ iso
• Sovellus tai KJ voi vaihdella sivun kokoa?
Tbl 8.2 [Stal05]
4

Table 8.2 Example Page Sizes

Computer	Page Size
Atlas	512 48-bit words
Honeywell-Multics	1024 36-bit word
IBM 370/XA and 370/ESA	4 Kbytes
VAX family	512 bytes
IBM AS/400	512 bytes
DEC Alpha	8 Kbytes
MIPS	4 kbyes to 16 Mbytes
UltraSPARC	8 Kbytes to 4 Mbytes
Pentium	4 Kbytes or 4 Mbytes
PowerPc	4 Kbytes
Itanium	4 Kbytes to 256 Mbytes

5

Sopiva sivukoko?

- Laitteisto (MMU) määrää mitä sivukokoa KJ:n käytettävä
  - sivukoko aina 2:sen potenssi
    - nopea osoitemuunnos
    - katkaisu ja katenointi helppoa
- Mitä isommat sivut, sitä vähemmän sivuja/prosessi
  - pienempi sivutaulu vie vähemmän tilaa
  - ison sivutaulun osia useammin levyllä
    - enemmän keskeytyksiä

6

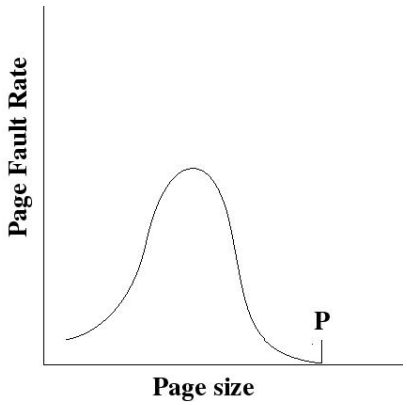
Sopiva sivukoko?

- Pieni: aiheuttaa väh. sisäistä pirstoutumista
- Iso: sisältää paljon esim. tarpeetonta koodia
- Iso: saattaa sopia paremmin yhteen levysiirtojen kanssa (lohkokoko)
  - hakuvarren siirto ja pyörähdysviive syö paljon aikaa
- Iso: viittaus useammin samalle sivulle
  - TLB:n osumatodennäköisyys hyvä

7

Sivupuutosten määrä

- Prosessille varatulle alueelle sopii enemmän pieniä sivuja kuin suuria
- Pieneltä sivulta viite usein muille sivuille, muistiin valikoituu pian ne sivut joita käytetään paljon
  è vähän sivupuutoksia
- Tn. että isolta sivulta viitattu sivu muistissa pienempi
  è paljon sivupuutoksia
- Kun sivukoko lähestyy prosessin kokoa,
  è vähän sivupuutoksia



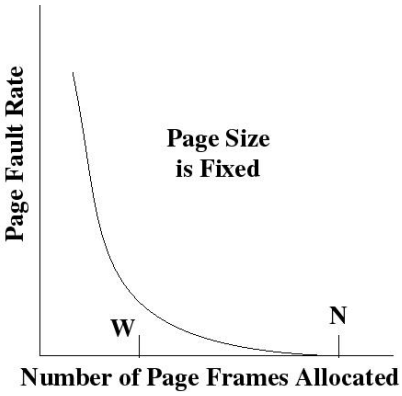
P=prosessin koko

8

Sivupuutosten määrä

- Paljonko sivutiloja per prosessi?
- Jos vähän sivutiloja, KJ heittää helposti pois sivun, jota tarvitaan pian uudelleen
  - è paljon sivupuutoksia
- Jos saa paljon, niin lähes kaikki sivut mahtuvat muistiin
  - è vähän sivupuutoksia

q lokaalit vs. globaalit algoritmit



W = working set size
N = total # of pages in process

9

Sopiva sivukoko?

- Yleisimmin käytetty sivukoko 4KB
  - ks. myös taulukko 8.2
- Tämä ei sovellu kaikkiin tarpeisiin, eräät prosessorit sallivat useita sivukokoja
  - Pentium sallii 2 kokoa: 4KB tai 4MB
  - MIPS peräti 7 eri koko: 4KB ..16MB
- ***Miten MMU tietää mitä kokoa käyttää?***

10

Optimaalinen sivun koko?

- Minimoi prosessille hukkatila (p. 238 [Tane01])
  - hukkatila = sivutaulun koko plus sisäinen pirstoutuminen?

$f = se/p + p/2$

s = keskim. prosessin koko tavuina
e = sivutaulu entryn koko
p = sivun koko
f = hukkatila

$df/dp = -se/p^2 + ½ = 0$, kun $p = \sqrt{2se}$

s = 1 MB e = 8B →	p_{opt} = 4 KB
s = 100 MB e = 8B →	p_{opt} = 40 KB

Tbl 8.2 [Stal05]

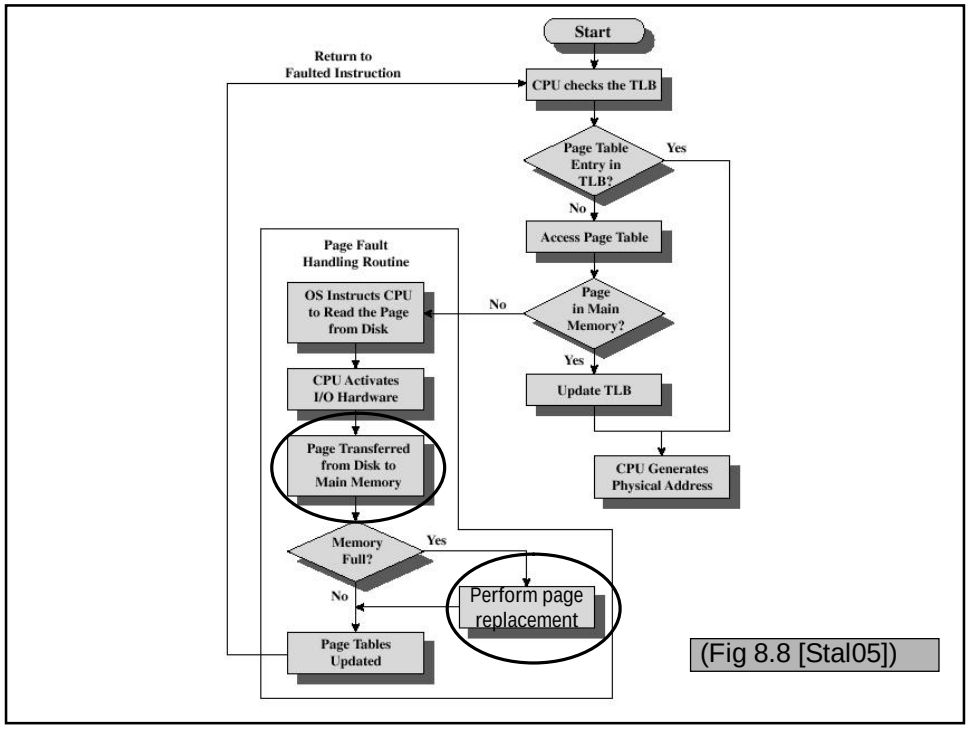
  - sovellukset suurempia ⟶ optimi sivun koko suurempi!
    - kriteerinä tässä vain hukkatilan määrä
    - mikä muu voisi olla kriteerinä?

11

Muistinhallinta-algoritmit

- Noutopolitiikka (fetch policy)
  - milloin sivu tuodaan muistiin?
- Sijoituspolitiikka (placement policy)
  - minne sivu sijoitetaan?
- Korvauspolitiikka, poistopolitiikka (replacement policy)
  - mikä sivu korvataan uudella ja milloin?
- Levylle kirjoitus -politiikka (cleaning policy)
  - milloin muutettu sivu vapautetaan?
- Moniajoaste (load control)
  - montako prosessia suoritettavana?
  - paljonko muistia käytettävissä per prosessi?
- Käyttöjoukon koko (working set)
  - paljonko sivukehyksiä per prosessi? (resident set)
  - mihin viitattu viime aikoina?

12



(Fig 8.8 [Stal05])

Milloin sivu muistiin levyltä?

- Tarvesivutus (demand paging)
  - tuo vasta kun viitataan sivulla olevaan osoitteeseen
  - aluksi paljon sivunpuutoksia, voi kestää kauan
    - voi olla hyvin huono juttu, esim. puhelinkeskus
  - hetken päästä paikallisuus alkaa olla OK
- Ennaltanouto (prepaging)
  - tuo etukäteen useita sivuja (tai kaikki?)
  - sivunpuutoksen sattuessa tuo monta sivua
    - alueellinen paikallisuus
  - peräkkäisiä sivuja ⟶ optimoi hakuviiveet
  - viitataanko tuotuun sivuun?
- Välimuotoja
  - tuo tärkeät ensin (demand)
  - tuo taustalla loput (prepaging)

Windows DLL epäsuora dynaaminen linkitys (implicit linking)

15

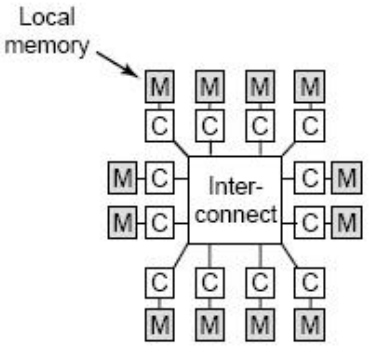
Sijoituspolitiikka

16

Minne sivu sijoitetaan?

- Kun noudetaan levyltä muistiin, ...
- Segmentointi
  - Best-fit, first-fit, ... (Ch 7)
- Sivutus ja segmentoiva sivutus
  - mikä tahansa vapaa sivutila
  - osoitemuunnos MMU:ssa aina yhtä tehokkaasti
- NUMA moniprosessorit
  - nonuniform memory access
  - yhteiskäytössä oleva muisti
  - tiedon fyysinen sijainti vaikuttaa muistiviitteen nopeuteen
  - sijoituspolitiikasta taas tärkeä?
  - sivu viitteen tekijälle vai omistajalle?

(Tane01, Fig 8-1)



17

Käyttöjärjestelmät II

Korvauspolitiikka

18

Muistitilan hallinta (resident set mgt)

- Montako sivutilaa maksimi per prosessi eli montako sivua pidetään muistissa? (resident set size)
  - kiinteä lkm?
    - miten iso? sama kaikilla?
  - vaihteleva lkm?
    - millä perusteella lkm vaihtelee?
    - käyttöjoukko (working set): "niiden sivujen joukko, jotka tarvitaan lähitulevaisuudessa"
      - resident set: "ne sivut, jotka ovat nyt muistissa"
- Mistä kaikkialta korvattavaa sivua etsitään?
  - etsi kaikkien sivujen joukosta (globaali politiikka)
  - etsi prosessin omien sivujen joukosta (lokaali politiikka)
    - voi olla silti vaihteleva sivukehysten lkm

19

Sivun korvaus

Katso Fig 8.1 [Stal05]

- Milloin korvataan?
  - sivunpuutos
  - muisti liian täynnä, ei vapaata tilaa (yleensä on!)
  - ei tarpeeksi vapaata tilaa
- Mikä sivu korvataan?
  - sellainen, jota ei tarvita (lähi)tulevaisuudessa
    - yritä ennustaa menneisyyden perusteella
  - paikallisuus: jos sivuun ei enää viitata, on ohjelma ohittanut ko. vaiheen
    - menneisyys ei ennustanutkaan tulevaisuutta oikein!

20

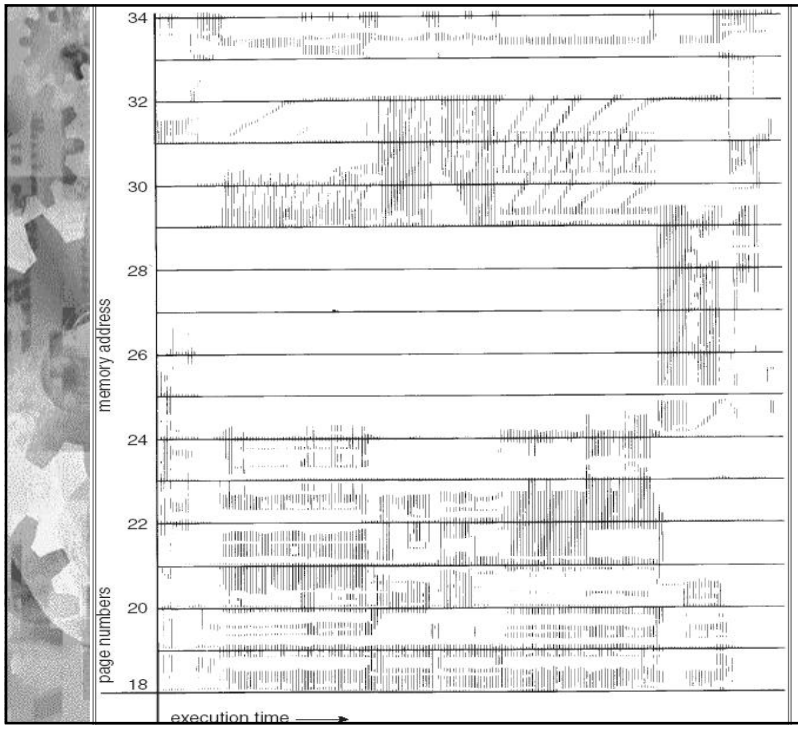


Fig 8.1 [Sta05]

Fig. 9.19 [SGG07]

21

Prosessille allokoidun muistin hallinta (Resident Set Management)

- Allokoitu muistin määrä (sivukehysten lkm)
  - jos liian pieni, niin
    - muistiin mahtuu useampi prosessi
    - enemmän sivunpuutoskeskeytyksiä
      - liikaa? trashing eli ruuhkautuminen?
  - jos turhan iso
    - vie tilaa muilta prosesseilta
    - CPU:n käyttöaste (utilisation) turhan pieni
    - ”tuhlattu” muistia

22

Muistiallokoinnin koko vs. poistettavan sivun valinta

	Local Replacement	Global Replacement
Fixed Allocation	•Number of frames allocated to process is fixed. •Page to be replaced is chosen from among the frames allocated to that process.	•Not possible.
Variable Allocation	•The number of frames allocated to a process may be changed from time to time, to maintain the working set of the process. (esim. PFF, kalvo 35) •Page to be replaced is chosen from among the frames allocated to that process.	•Page to be replaced is chosen from all available frames in main memory; this causes the size of the resident set of processes to vary.

Tbl 8.4 [Stal05]

23

Lukitus

- Sivu voidaan lukita muistiin
  - ytimeen kuuluvat sivut
  - KJ:n keskeiset tietorakenteet
  - siirrännän puskurit (siirron ajaksi)
  - reaaliaikajärjestelmissä myös prosessin sivut
- Toteutus
  - sivutilataulussa ko. sivutilan kohdalla lukkobitti
    - vapaat / varatut sivutilat
  - tai sivutaulun alkiossa lukkobitti
    - tietylle prosessille varatut sivutilat
  - globaali vs. lokaali politiikka

24

Korvauspolitiikka

- Optimaalinen algoritmi (OPT) korvaa sivun, johon ei tule enää viittauksia, tai sivun, johon uudelleenviittaamiseen menee pisin aika
  - mistäpäs sen tiedät?
- Perusalgoritmit — on muitakin…
  - LRU, Least recently used
    - poista se, johon viitattu kauimmin aikaa sitten
  - FIFO, First-in-first-out
    - poista se, joka ollut kauimmin muistissa
  - Clock
    - poista se, jota ei käytetty edellisen tutkinnan jälkeen
    - 'rengasalgoritmi'
- Hyvyyden mitta?
  - sivunpuutoskeskeytysten lukumäärä (tiheys)

25

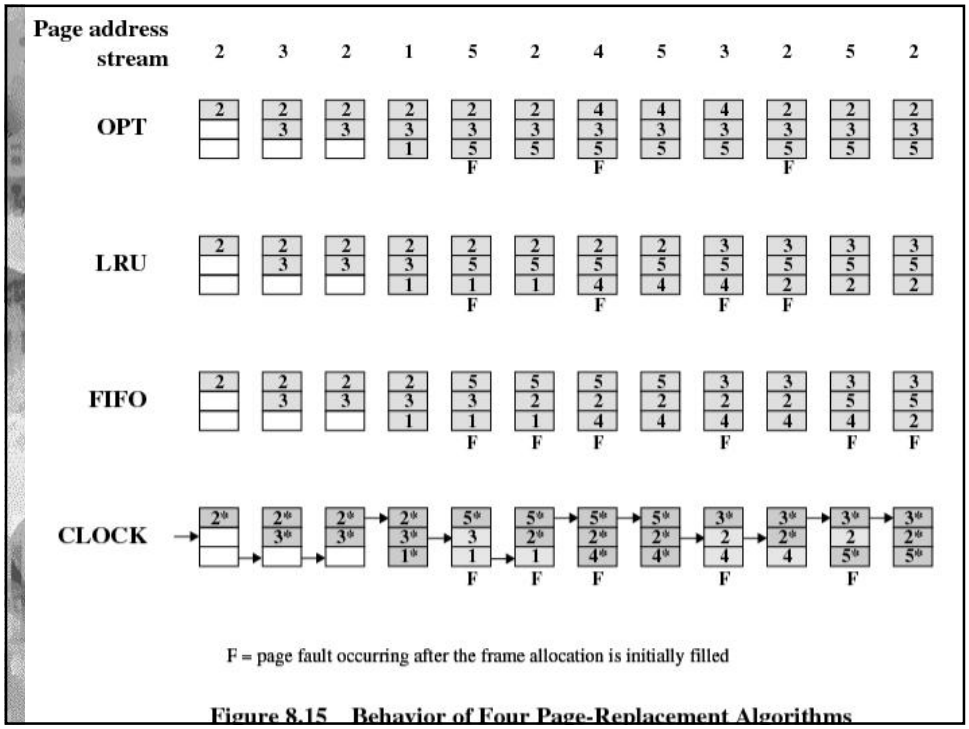


Figure 8.15 Behavior of Four Page-Replacement Algorithms

OPT-poistoalgoritmi

Fig 8.15 [Stal05]

- Poista se, jonka seuraavaan viittaukseen on eniten aikaa
  - vaatii erinomaisia ennustajan taitoja
  - voidaan mitata jälkeenpäin trace-tiedostosta
- Jälkiviisaus on hienoa, mutta mitä sillä tekee?
- Vertailukohta poistoalgoritmeja arvioitaessa
  - jos päästään lähelle OPTia, niin ollaan jo aika hyviä
  - jos ollaan kaukana OPTista, niin huonosti menee

27

RANDOM poistoalgoritmi

- Poista satunnainen sivu
- Todella helppo toteuttaa
- Ei vaadi laitteistotukea
- Lähes yhtä hyvä kuin FIFO
  - ei siis kauhean hyvä…

28

LRU poistoalgoritmi

- Korvaa sivu, johon viittaamisesta kulunut kauimmin menneisyydessä
  - vrt OPT: *... kauimmin tulevaisuudessa*
- Paikallisuus vihjaa tähän suuntaan
- Toteutuksen vaikeus
  - jokaiseen sivuun mukaan aikaleimako?
  - miten aikaleima päivitetään?
    - jokaisella muistiviitteellä
    - laitteistotoimintona? miten?
  - ei käy, liikaa yleisrasitetta!

Fig 8.15 [Stal05]

29

NRU (Not Recently Used) - poistoalgoritmi

- Sivukehyksissä viitebitti (R, Reference)
  - nollataan aika ajoin
  - laitteisto asettaa, jos viittaus

vaatii siis laitteistotukea!

- Sivukehyksessä muutettu bitti (M, Modified)
  - nollataan levylle kirjoituksen yhteydessä
  - laitteisto asettaa, jos kirjoitus
    - tarvitaan siis joka tapauksessa!
- Poista
  - joku, johon ei viittausta ja jota ei muutettu
  - joku, johon ei viittausta ja jota muutettu
  - joku, johon viittaus ja jota ei muutettu
  - joku
- Helppo toteuttaa, vähän HW-tukea

30

FIFO-poistoalgoritmi

Fig 8.15 [Stal05]

- Korvaa sivu, joka ollut kauimmin muistissa
- vrt. Round-Robin vuorottamisessa
- Helpoin toteuttaa
  - aikaleima sivutaulun alkioon, kun tuodaan muistiin, TAI
  - kaikki sivut listassa, muistiin tuontiajan mukaisesti
- Muistiin tuontiaika korreloi huonosti käyttötarpeen kanssa
  - kauiten muistissa ollut sivu voi olla sellainen, jota käytetään koko ajan?

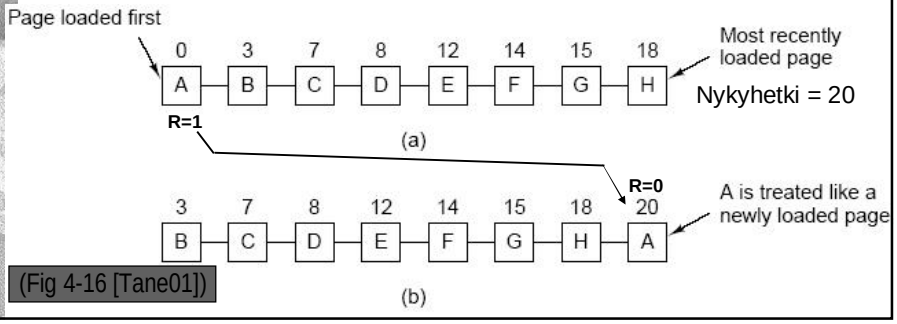
jonojen hallinta?

31

Second Chance -poistoalgoritmi

- FIFOn modifikaatio
- viitebitti R ja muutosbitti M
- poista FIFO-järjestyksessä, mutta
  - jos R==1 (on viitattu äsken), niin siirrä FIFO jonon loppuun ja nollaa R
    - jos muita ei löydy, niin sitten tämä poistetaan kun FIFO jonossa päästään seuraavalla kerralla tämän sivun kohdalle (jolloin R==0)

jonojen hallinta?



(Fig 4-16 [Tane01])

Clock-poistoalgoritmi

- Second Chance –algoritmin modifikaatio
  - ei sivujen siirtelyä listassa
- Käy sivutilojen listaa läpi renkaana
- Jokaiseen sivutilaan liittyy viitebitti (U eli Use bit)
  - kun sivu muistiin, U = 0, reset silloin tällöin
  - kun sivuun viitataan, U = 1 (MMU asettaa)
- Korvaa sivu, jonka viitebitti on 0
  - ensimmäinen eteen sattuva
- Aseta aina tutkituissa viitebitiksi 0
  - jos arvo seuraavallakin kierroksella 0, sivuun "ei ole viitattu" (ainakaan teoriassa)
- Aseta kaikki U=0
  - ei aina? joka 10 ms? (HW-toiminto)

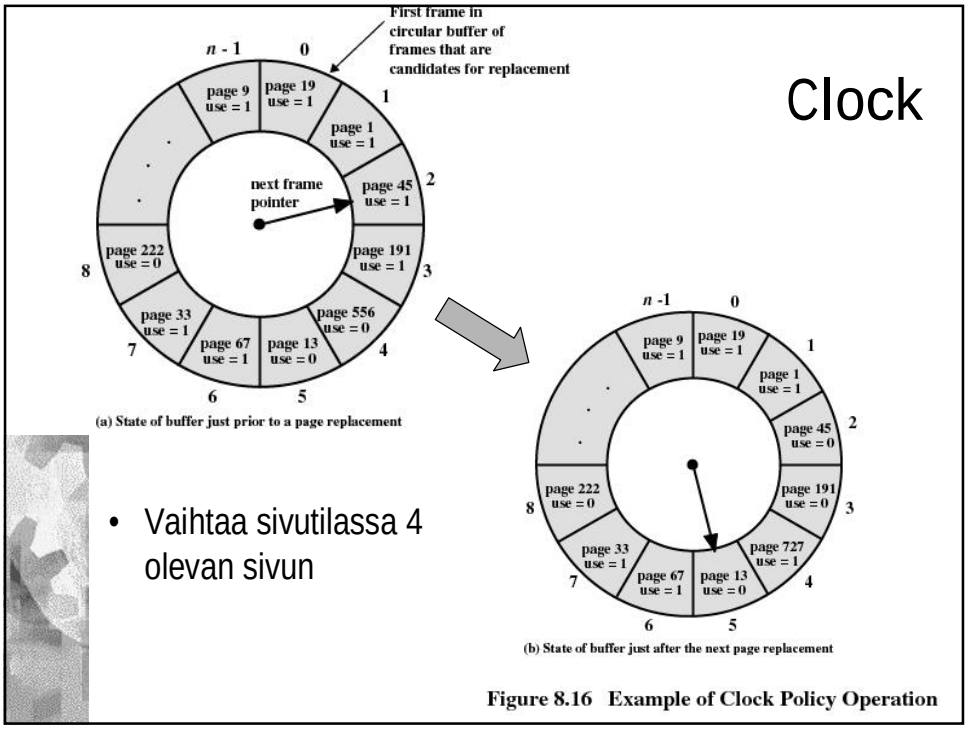
Kts Fig 8.16 [Stal05]

Kts Fig 8.15 [Stal05]

Kts Fig 8.17 [Stal05]

33

Clock



- Vaihtaa sivutilassa 4 olevan sivun

Figure 8.16 Example of Clock Policy Operation

Vertailua

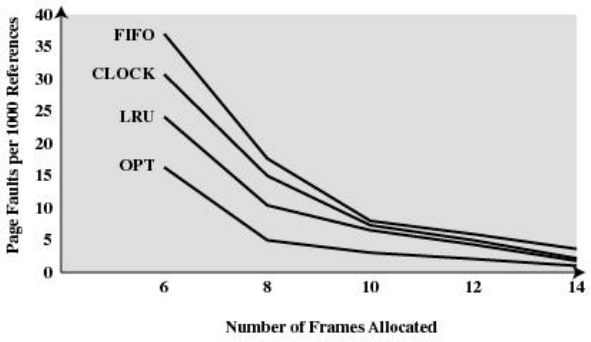


Figure 8.17 Comparison of Fixed-Allocation, Local Page Replacement Algorithms

35

Clock-poistoalgoritmi M-bitin kera

- Lisää laitteistotukea: M eli Modified-bitti
  - ennestään U (eli Used eli R eli Referenced)
  - M-bitti tarvitaan joka tapauksessa
- käy sivuja läpi ringissä
  - jos U==0 ja M==0, poista tämä
  - älä muuta U-bittiä
- käy sivuja läpi uudestaan ringissä
  - jos U==0 ja M==1, poista tämä
  - aseta U=0 muille
- käy sivuja läpi uudestaan ringissä
  - jos löytyy M==0, niin poista tämä
- käy sivuja läpi uudestaan ringissä
  - jos löytyy M==1, niin poista tämä
  - kaikki jäljellä olevat tällaisia

Not used, not modified?

Not used, modified?

Not modified?

Any page?

käy rinki läpi monta kertaa, kunnes poistettava löytynyt

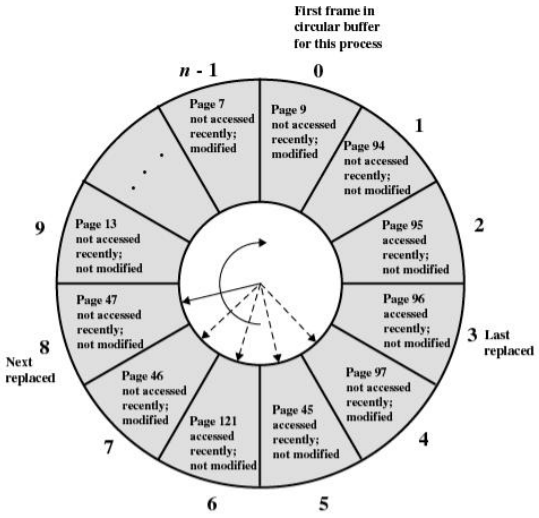
Kts. Fig 8.18 [Stal05]

Monta läpikäyntiä hidasta, jos suuri muisti eli iso rinki

36

Clock-poistoalgoritmi

Fig 8.16 [Stal05]



37

Käyttöjoukkostrategia

Muistitilan koon hallintaan

38

Käyttöjoukkostrategia (Working Set Strategy)

- Sivut, joihin viitattu k:n (ikkunakoko) viimeisimmän viittauksen aikana

  Kts Fig 8.19 [Stal05]

  - approksimaatio prosessin tarvitsemille sivuille (todelliselle käyttöjoukolle)
- Toteutus? $W(t, \Delta)$

  ”working set” vs. ”resident set”?
  - vain approksimaatioita
  - käyttöjoukossa sivut, joihin viitattu esim. viim. 100 ms:n aikana, kun prosessi oli suoritettavana
- Sivu, joka ei kuulu käyttöjoukkoon, voidaan vapauttaa
  - pyrkii tekemään ennakolta tilaa uusille sivuille
- Jos käyttöjoukko kutistuu pieneksi, voidaan tuoda sivuja ennalta takaisin
  - mitkä? viitattu ja sitä seuraava?

39

Esimerkki: Käyttöjoukko

Fig 8.19 [Stal05]

Sequence of Page References	Window Size, Δ			
	2	3	4	5
24	24	24	24	24
15	24 15	24 15	24 15	24 15
18	15 18	24 15 18	24 15 18	24 15 18
23	18 23	15 18 23	24 15 18 23	24 15 18 23
24	23 24	18 23 24	•	•
17	24 17	23 24 17	18 23 24 17	15 18 23 24 17
18	17 18	24 17 18	•	18 23 24 17
24	18 24	•	24 17 18	•
18	•	18 24	•	24 17 18
17	18 17	24 18 17	•	•
17	17	18 17	•	•
15	17 15	17 15	18 17 15	24 18 17 15
24	15 24	17 15 24	17 15 24	•
17	24 17	•	•	17 15 24
24	•	24 17	•	•
18	24 18	17 24 18	17 24 18	15 17 24 18

40

Käyttöjoukon koko

- Mikä sopiva käyttöjoukon koko?
  - arvio esim. sivupuutostiheyden perusteella
- Montako sivutilaa siis per prosessi?
  - osa käyttöjoukoissa, osa pidettävä vapaana
  - vähän ⟶ muistissa monta prosessia, usein puutoksia
  - paljon ⟶ vähemmän sivunpuutoskeskeytyksiä
  - paikallisuus vs. sivunpuutos
- Kiinteä allokointi
  - prosessia käynnistettäessä
  - tyypin perusteella: interaktiivinen, erätyö (tausta)
  - muun tiedon perusteella, esim. edellinen suorituskerta
- Dynaaminen allokointi
  - sivuja muistiin tarpeen mukaan
  - käyttöjoukon koko vaihtelee vapaan muistitilan mukaan ja prosessin vaiheen mukaan

$$1 \leq |W(t, \Delta)| \leq \min(\Delta, n)$$

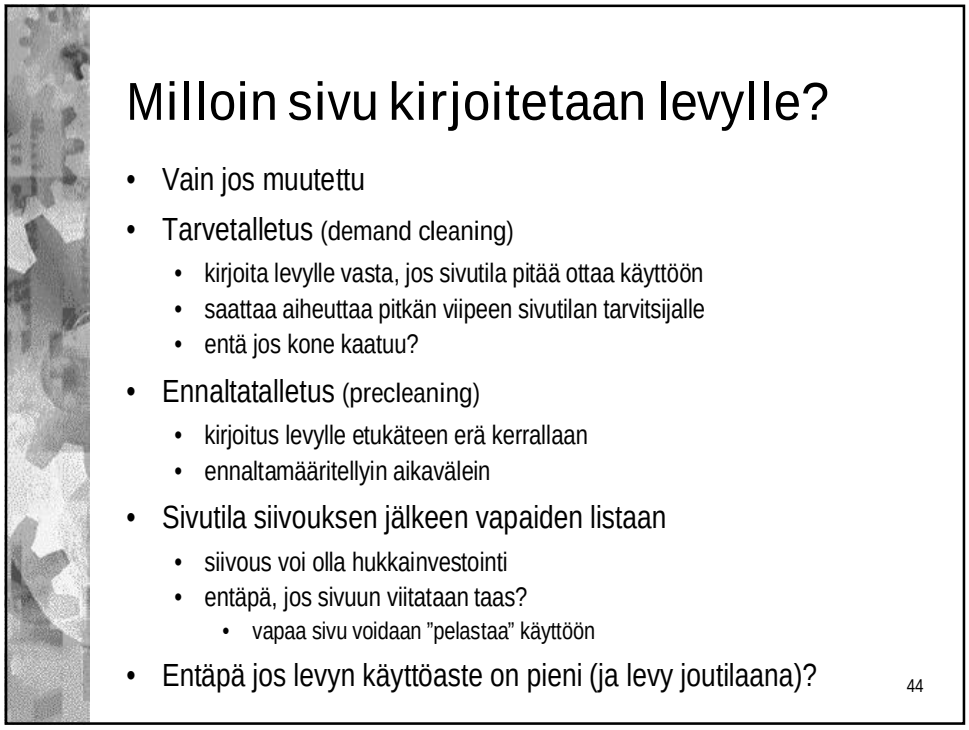
41

Dynaaminen käyttöjoukon koko

- PFF – Page Fault Frequency
  - pidetään kirjaa, milloin edellinen sivunpuutos tapahtui
  - sivunpuutos: laske T = väliaika edellisestä sivunpuutoksesta
  - jos T < L, niin tarvitaan lisää muistitilaa
    - lisää puuttuva sivu käyttöjoukkoon
  - jos T > H, niin prosessilla ”liikaa” muistitilaa
    - poista kaikki viime aikoina viittaamattomat sivut tai jotkut niistä käyttöjoukosta
  - yritetään pitää sivunpuutostiheys sopivissa rajoissa, eli sivunpuutosten väliaika välillä (L, H)
- VSWS – Variable-interval Sampled Working Set
  - PFF:n modifikaatio, ottaa huomioon vaiheen vaihtumisen
  - käyttöjoukko voi kasvaa milloin vain, mutta pienentyä vain vaiheen vaihtuessa
  - vaiheen pituus aina vähintään M ja enintään L
  - vaiheen aikana hyväksyttävän sivunpuutosmäärän (Q) avulla
  - vaihe vaihtuu hetkellä L, tai kun Q sivunpuutosta (mutta ei ennen hetkeä M)

42


Levylle kirjoitus –politiikka
(Cleaning Policy)
”how to clean dirty pages”
43


Milloin sivu kirjoitetaan levylle?
• Vain jos muutettu
• Tarvetalletus (demand cleaning)
• kirjoita levylle vasta, jos sivutila pitää ottaa käyttöön
• saattaa aiheuttaa pitkän viipeen sivutilan tarvitsijalle
• entä jos kone kaatuu?
• Ennaltatalletus (precleaning)
• kirjoitus levylle etukäteen erä kerrallaan
• ennaltamääritellyin aikavälein
• Sivutila siivouksen jälkeen vapaiden listaan
• siivous voi olla hukkainvestointi
• entäpä, jos sivuun viitataan taas?
• vapaa sivu voidaan ”pelastaa” käyttöön
• Entäpä jos levyn käyttöaste on pieni (ja levy joutilaana)?
44

Puskurointi (page buffering)

- Pidä tietty osa sivutiloista vapaana
  - joskus tulee vapautetuksi väärä sivu

~ sivujen välimuisti (äskettäin vapautetut sivut)

- nopea palautus takaisin käyttöön sivupuskurista, jos ”vapaassa” sivukehyksessä olevaan tietoon viitataan

- Poistettavaksi merkitty sivu lisätään
  - vapaiden listaan, jos ei muutettu
  - muutettujen sivujen listaan, jos muutettu
- Sivu pysyy silti alkuperäisellä paikalla muistissa
  - vain merkintä prosessin sivutaulussa poistetaan
- Varaus vapaiden listan alusta
  - viimeksi vapautetuilla suurempi mahdollisuus tulla ”pelastetuksi”
    - miksi?

45

Moniajoaste

46

Moniajoaste

- Löydettävä sopiva suorituksessa (muistissa) olevien prosessien lkm
- Liian vähän?
  - CPU jouten, jos prosessit Blocked tilassa
- Liian paljon?
  - ruuhkautuminen (trashing): sivuttaminen vie liikaa aikaa

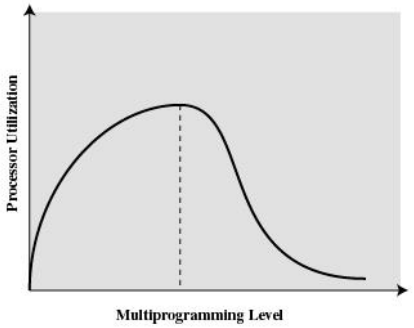


Figure 8.21 Multiprogramming Effects

Fig 8.21 [Stal05]

47

Moniajoaste

- Mitä tehdä jos liikaa prosesseja?
  Eli siis liian vähän muistia per prosessi?

- Heittovaihda Ready-prosesseja levylle
  - se, jolla pienin prioriteetti
  - se, joka aiheutti puutoksen
  - se, joka saapui viimeisenä (=uusin)
  - se, jolla pienin käyttöjoukko
  - se, jolla eniten muistia käytössä tai jolla suurin aikaviipale
  - kuka vaan, mutta joku!

48

Moniajoasteen dynaaminen kontrolli

- L=S –kriteeri
  - sivunpuutosten väliaika L
  - sivunpuutosten käsittelyaika S
  - L >> S ⟶ kasvata mpl, L << S ⟶ pienennä mpl
    - mikä on ”paljon”?
- 50% sääntö
  - Util (sivutuslevy) olisi hyvä olla 50%
  - ”pieni” U ⟶ kasvata mpl, ”iso” U ⟶ pienennä mpl
- Globaali Clock poistoalgoritmi, viisarin nopeusrajat
  - ”hidas” ⟶ kasvata mpl
  - ”nopea” ⟶ pienennä mpl

49

Heittovaihtoalue

50

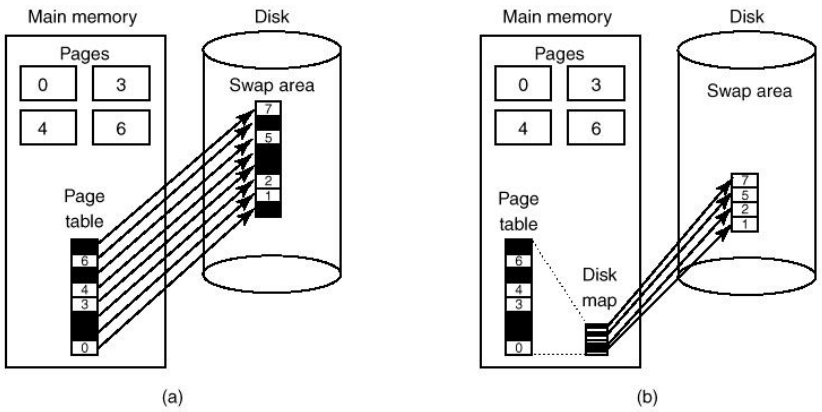
Heittovaihtoalue (swap area, VM backup store)

- Heittovaihtoalue varattu levyltä etukäteen
  - Windows: pagefile.sys, win386.swp
  - Linux: oma swap-levypartitio
- a) Kiinteä yhtenäinen varaus
  - kerralla koko prosessille
    - kopioi koodi alustuksessa tai
    - varaa tila, heittovaihda sivut sinne vähitellen
  - PCB:ssa swap-alueen alkuosoite ja pituus
  - vapaiden alueiden kirjanpito kuten luvussa 7
- b) Dynaaminen varaus
  - sivu kerrallaan
  - tarvitaan sivukohtainen kuvaus
    - sivutaulun alkiossa swap-alueen lohkonnumero tai
    - erillinen taulu (disk map)
  - ei tarvita levytilaa sivuille, joita ei koskaan viedä levylle

51

Heittovaihtoalue

(Fig 4-33 [Tane01])



Kiinteä varaus

Dynaaminen varaus

52