LUENTO 5

Käyttöjärjestelmät II

MUISTINHALLINNAN OHJELMISTO

Ch 7-8 [Stal 05]

Lähtötiedot / seuraavaksi?

Yksinkertainen muistinhallinta (Ch 7)

- **Tehtävät, staattinen vs. dynaaminen partitiointi, Buddy System sivutus, segmentointi**

Virtuaalimuisti (Ch 8.1)

- **Sivutus ja segmentointi, sivu- ja segm. taulut, MMU**

Laitteistotuki (Tietokoneen rakenne -kurssi)

- **MMU:n sisäinen rakenne**
- **TLB**

Seuraavaksi KJ-II:ssa

- **KJ:n muistinhallinta-algoritmit** (Ch 8.2) — Luento 5
- **UNIX ja Solaris: muistinhallinta** (Ch 8.3) — Luento 6
- **Linux: muistinhallinta** (Ch 8.4)
- **W2K: muistinhallinta** (Ch 8.5)

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**

Looginen vs. fyysinen osoite

1502 → 6766

(Fig 7.12 [Stal05])

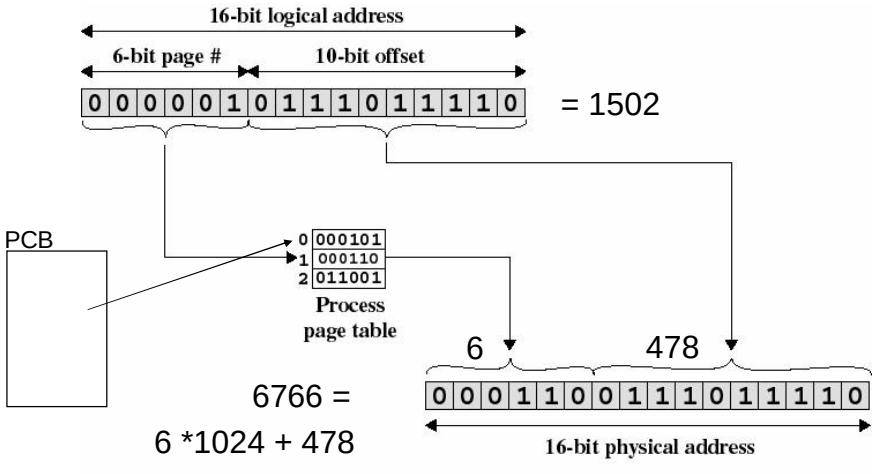
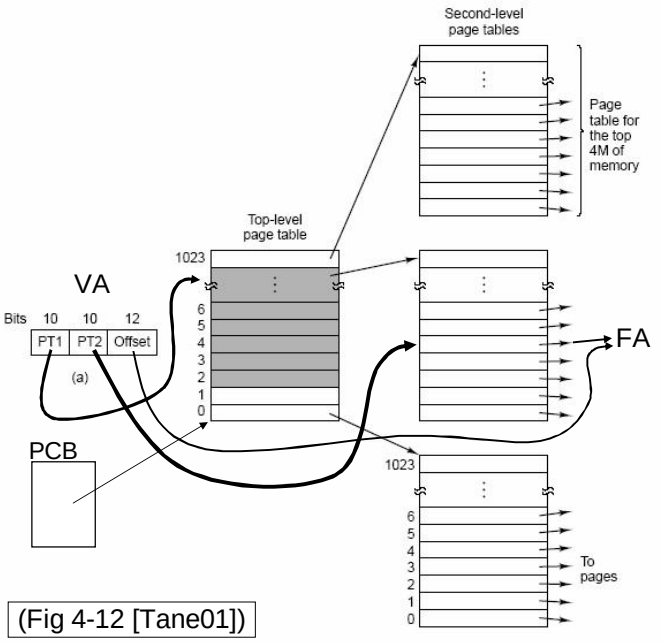


Fig 8.13 [Stal05]

Moni-tasoinen sivutus



(Fig 4-12 [Tane01])

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]

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

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

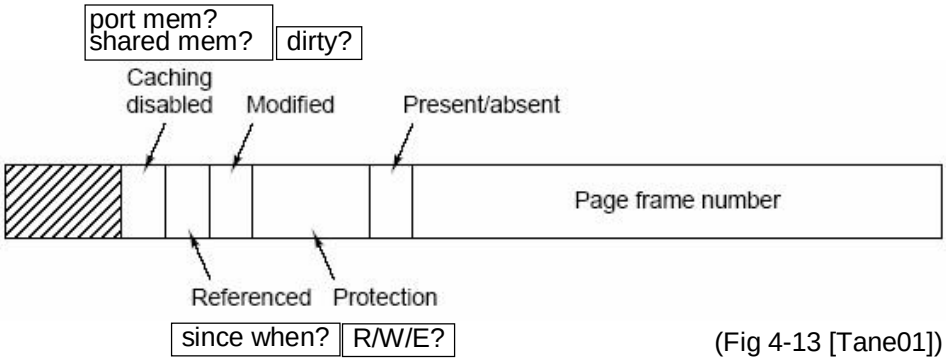
s = 1 MB e = 8B	p_{opt} = 4 KB
s = 100 MB e = 8B	p_{opt} = 40 KB

Tbl 8.2 [Stal05]

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

Sivutaulun alkio

- **Jokaisella prosessilla oma sivutaulunsa**
  - missä sivukehyksissä tämän prosessin sivut sijaitsevat?
  - sivutaulun fyysinen osoite PCB:ssä



(Fig 4-13 [Tane01])

TLB

- **Miten osoitteen muutos nopeasti yleensä?** Fig 8.7 [Stal05]
- **Samalla teknologialla kuin (tason 1) välimuisti?** Fig 8.10 [Stal05]
  - vain ajallinen paikallisuus
- **Hudin käsittely laitteistolla tai TLB-keskeytyksellä**

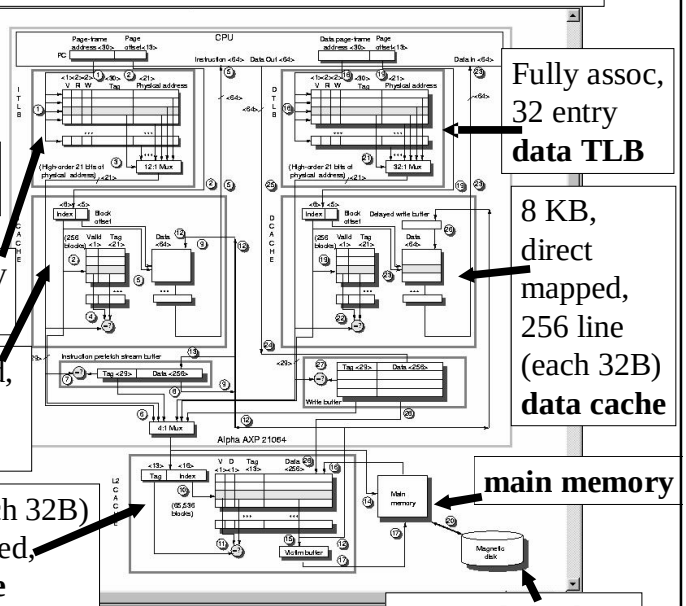
Valid	Virtual page	Modified	Protection	Page frame
1	140	1	RW	31
1	20	0	R X	38
1	130	1	RW	29
1	129	1	RW	62
1	19	0	R X	50
1	21	0	R X	45
1	860	1	RW	14
1	861	1	RW	75

Fig. 4-14. A TLB to speed up paging. [Tane01]

TLB tarkemmin? ⟶ Tietokoneen rakenne -kurssi

Fig. 5.47 from Hennessy-Patterson, Computer Architecture

DEC Alpha AXP 21064 memory hierarchy



Fully assoc, 32 entry **data TLB**

8 KB, direct mapped, 256 line (each 32B) **data cache**

main memory

Fully assoc, 12 entry **instruction TLB**

8 KB, direct mapped, 256 line (each 32B) **instruction cache**

2 MB, 64K line (each 32B) direct mapped, unified, write-back **L2 cache**

paging **disk** (dma)

Muistinhallinta-algoritmit

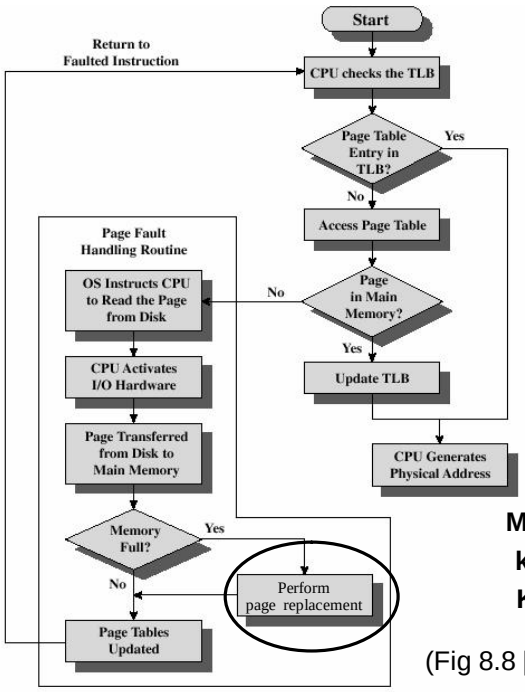
- **Noutopolitiikka** (fetch policy)
  - milloin sivu tuodaan muistiin?
- **Sijoituspolitiikka** (placement policy)
  - minne sivu sijoitetaan?

Tbl 8.3 [Stal05]

- **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?

Käyttöjärjestelmät II

Noutopolitiikka



Muut osat käsitelty KJ-I:ssä

(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)

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola 22.3.2006 14

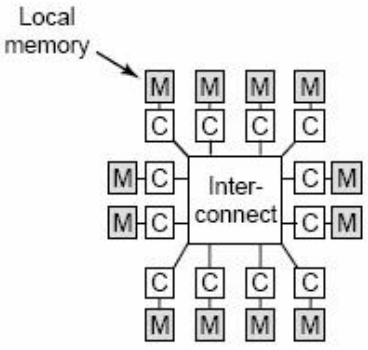
Käyttöjärjestelmät II

Sijoituspolitiikka

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**
  - **n**on**u**niform **m**emory **a**ccess
  - yhteiskäytössä oleva muisti
  - tiedon fyysinen sijainti vaikuttaa muistiviitteen nopeuteen
  - sijoituspolitiikasta taas tärkeä?
  - sivu viitteen tekijälle vai omistajalle?

(Tane01, Fig 8-1)



Käyttöjärjestelmät II

Korvauspolitiikka

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

Sivun korvaus

- **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!

Fig 8.1 [Stal05]

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 utilisaatio turhan pieni
    - ”tuhlattu” muistia

Fig 8.1 [Stal05]

Fig 8.11 (b) [Stal05]

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])

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

Korvauspolitiikka

- **Optimaalinen algoritmi (OPT) korvaa sivun, johon ei tule enää viittauksia, tai sivun, johon uudelleenviittaamiseen menee pisin aika**
  - mistäpäs sen tiedät?
- **Perusalgoritmit**
  - 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)

on muitakin…

OPT-poistoalgoritmi

- **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

Fig 8.15 [Stal05]

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]

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**

FIFO-poistoalgoritmi

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

Fig 8.15 [Stal05]

jonojen hallinta?

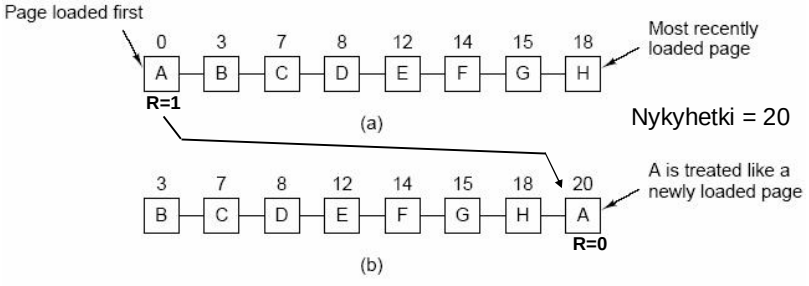
RANDOM poistoalgoritmi

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

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

Fig 8.16 [Stal05]

- **Aseta aina tutkituissa viitebitiksi 0**
  - jos arvo seuraavallakin kierroksella 0, sivuun “ei ole viitattu” (ainakaan teoriassa)

Fig 8.15 [Stal05]

Fig 8.17 [Stal05]

- **Aseta kaikki U=0**
  - ei aina? joka 10 ms? (HW-toiminto)

Clock-poistoalgoritmi M-bitin kera

- **Lisää laitteistotukea: M eli Modified-bitti**
  - ennestään U (eli Used eli R eli Referenced)
- **käy sivuja läpi ringissä**

Not used, not modified?

  - jos U==0 ja M==0, poista tämä
  - älä muuta U-bittiä
- **käy sivuja läpi uudestaan ringissä**

Not used, modified?

  - jos U==0 ja M==1, poista tämä
  - aseta U=0 muille
- **käy sivuja läpi uudestaan ringissä**

Not modified?

  - jos löytyy M==0, niin poista tämä
- **käy sivuja läpi uudestaan ringissä**

Any page?

  - jos löytyy M==1, niin poista tämä
  - kaikki jäljellä olevat tällaisia

käy rinki läpi monta kertaa, kunnes poistettava löytynyt

Fig 8.18 [Stal05]

Käyttöjärjestelmät II

Käyttöjoukkostrategia

Muistitilan koon hallintaan

Käyttöjoukkostrategia (Working Set Strategy)

Fig 8.1 [Stal05]

- **Sivut, joihin viitattu k:n** (ikkunakoko) **viimeisimmän viittauksen aikana**
  - approksimaatio prosessin tarvitsemille sivuille (todelliselle käyttöjoukolle)

Fig 8.19 [Stal05]

”working set” vs. ”resident set”?

- **Toteutus?**
  - 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?

Käyttöjoukon koko

- **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
- **Mikä sopiva käyttöjoukon koko?**
  - arvio esim. sivupuutostiheyden perusteella

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

Käyttöjärjestelmät II

Levylle kirjoitus –politiikka (Cleaning Policy)

”how to clean dirty pages”

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?
- **Entäpä jos levy odottaa toimetonna, eli sillä on pieni utilisaatio?**

Puskurointi (page buffering)

- Pidä tietty osa sivutiloista vapaana
  - joskus tulee vapautetuksi väärä sivu
- ~ sivujen välimuisti
  - 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?

Käyttöjärjestelmät II

Moniajoaste

Moniajoaste (load control)

- **Löydettävä sopiva suorituksessa (muistissa) olevien prosessien lkm**
- **Liian vähän?**
  - CPU jouten, jos prosessit Blocked tilassa

Fig 8.11 [Stal05]

- **Liian paljon?**

Fig 8.21 [Stal05]

  - ruuhkautuminen (trashing): sivuttaminen vie liikaa aikaa
- **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 oli edellisenä suorituksessa
    - se, jolla pienin käyttöjoukko
    - se, jolla eniten muistia käytössä tai jolla suurin aikaviipale
    - kuka vaan, mutta joku!

Moniajoasteen dynaaminen kontrolli

- **L=S –kriteeri**
  - sivunpuutosten väliaika L
  - sivunpuutosten käsittelyaika S
  - L >> S ⟶ kasvata mpl, L << S ⟶ pienennä mpl
- **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

Käyttöjärjestelmät II

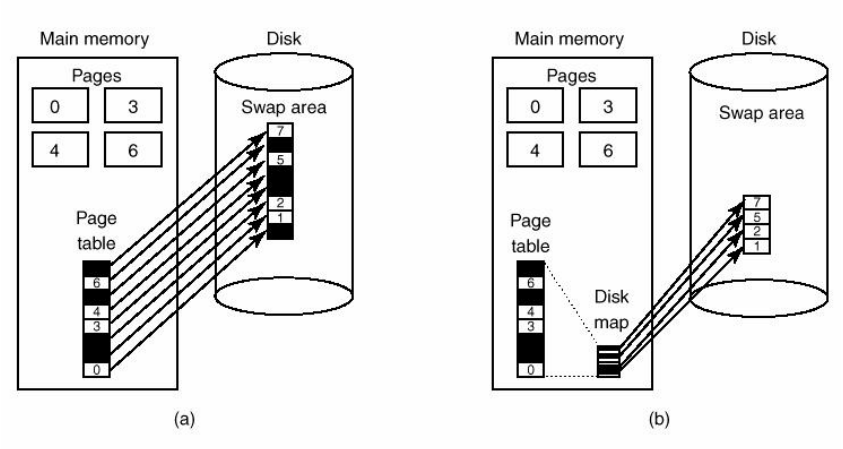
Heittovaihtoalue

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

Heittovaihtoalue

(Fig 4-33 [Tane01])



Kiinteä varaus

Dynaaminen varaus

Kertauskysymyksiä

- Miksi PFF:ssä kannattaa olla kaksi rajaa eikä vain yksi?
- Miten OPT voidaan käytännössä toteuttaa?
- Mikä on OPT:n merkitys?
- Mitä hyötyä kotikoneessa on suuresta moniajoasteesta?
- Miksi Clock-algoritmi asettaa ohitettaville sivuille use=0?
- Miksei sijoituspolitiikka ole niin kiinnostava?
- Mitkä hyötyä on vapaiden sivujen puskuroinnista? Miten sen aiheuttamia haittoja voidaan minimoida?

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola 22.3.2006 45