

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

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

Perusvalinnat

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

Looginen vs. fyysinen osoite

1502 → 6766 (Fig 7.12 [Stal05])

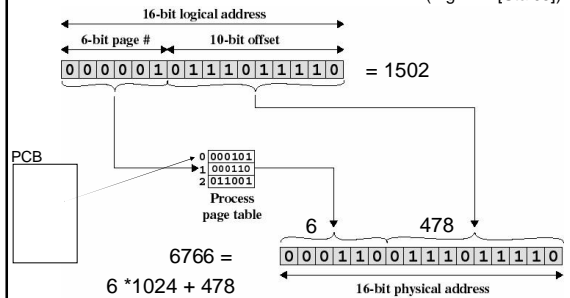
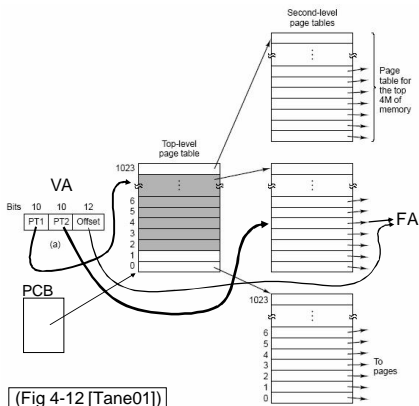


Fig 8.13 [Stal05]

Moni-tasoinen sivutus



(Fig 4-12 [Tane01])

Sivun koko

- n Optimoit sisäistä pirstoutumista → pieni
- n Optimoit sivutaulun (taulujen) kokoa → iso
- n Monikerta (1x, 2x, ...) levylohkon koosta
- n Optimaaliarvo erilainen eri ohjelmille
- n Optimoit TLB:n osumasuhdetta → iso
- n 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

$$df/dp = -se/p^2 + 1/2 = 0, \text{ kun } p = \sqrt{2se}$$

$$s = 1 \text{ MB } e = 8 \text{ B} \rightarrow p_{\text{opt}} = 4 \text{ KB}$$

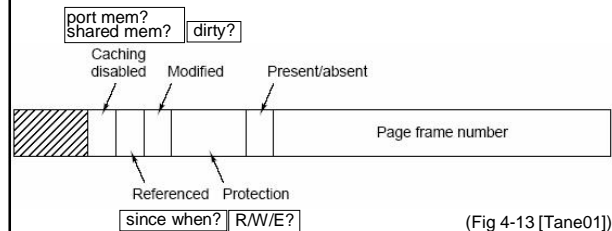
$$s = 100 \text{ MB } e = 8 \text{ B} \rightarrow p_{\text{opt}} = 40 \text{ KB}$$

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])

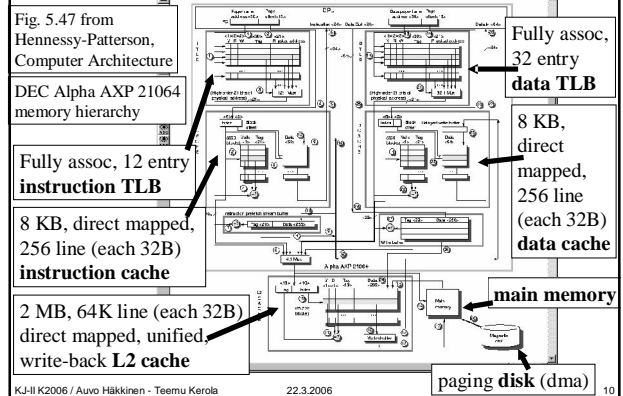
TLB

- 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

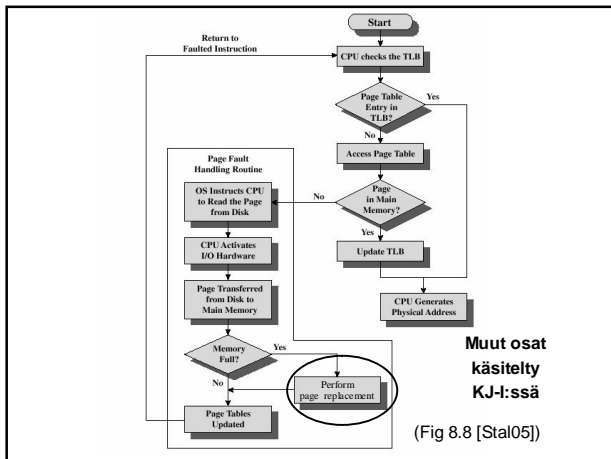


Muistinhallinta-algoritmit

- Noutopolitiikka** (fetch policy)
 - milloin sivu tuodaan muistiin?
- Sijoituspolitiikka** (placement policy) (Tbl 8.3 [Stal05])
 - minne sivu sijoitetaan?
- Korvauspolitiikka, poistopolitiikka** (replacement policy)
 - mikä sivu korvataan uudella ja milloin?
- Levyille 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



Milloin sivu muistiin levytä?

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

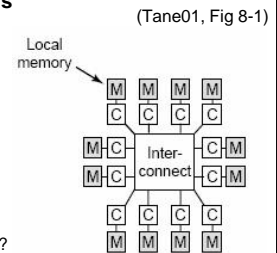
Windows DLL epäsuora dynaaminen linkitys (implicit linking)

Käyttöjärjestelmät II

Sijoituspolitiikka

Minne sivu sijoitetaan?

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



Käyttöjärjestelmät II

Korvauspolitiikka

Muistitilan hallinta (resident set mgt)

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

Sivun korvaus

- n **Milloin korvataan?**
 - u sivunpuutos
 - u muisti liian täynnä, ei vapaata tilaa (yleensä on!)
 - u ei tarpeeksi vapaata tilaa
- n **Mikä sivu korvataan?**
 - u sellainen, jota ei tarvita (lähi)tulevaisuudessa
 - F yritä ennustaa menneisyyden perusteella
 - u paikallisuus: jos sivuun ei enää viitata, on ohjelma ohittanut ko. vaiheen
 - F menneisyys ei ennustanutkaan Fig 8.1 [Stal05] tulevaisuutta oikein!

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

19

Prosessille allokoitun muistin hallinta (Resident Set Management)

- n **Allokoitu muistin määrä (sivukehysten lkm)**
 - u jos liian pieni, niin
 - F muistiin mahtuu useampi prosessi Fig 8.1 [Stal05]
 - F enemmän sivunpuutoskeskeytyksiä
 - liikaa? trashing eli ruuhkautuminen?
 - u jos turhan iso
 - F vie tilaa muilta prosesseilta Fig 8.11 (b) [Stal05]
 - F CPU utilisaatio turhan pieni
 - F "tuhlattu" muistia

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

20

Muistiallokoinnin koko vs. poistettavan sivun valinta

	Local Replacement	Global Replacement
Fixed Allocation	<ul style="list-style-type: none"> • Number of frames allocated to process is fixed. • Page to be replaced is chosen from among the frames allocated to that process. 	<ul style="list-style-type: none"> • Not possible.
Variable Allocation	<ul style="list-style-type: none"> • 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. 	<ul style="list-style-type: none"> • 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])

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

21

Lukitus

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

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

22

Korvauspolitiikka

- n **Optimaalinen algoritmi (OPT) korvaa sivun, johon ei tule enää viittauksia, tai sivun, johon uudelleenviittamiseen menee pisin aika**
 - u mistäpäsen sen tiedät?
- n **Perusalgoritmit** on muitakin...
 - u LRU, Least recently used
 - F poista se, johon viitattu kauimmin aikaa sitten
 - u FIFO, First-in-first-out
 - F poista se, joka ollut kauimmin muistissa
 - u Clock
 - F poista se, jota ei käytetty edellisen tutkimuksen jälkeen
 - F 'rengsalgoritmi'
- n **Hyvyyden mitta?**
 - u sivunpuutoskeskeytysten lukumäärä (tiheys)

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

23

OPT-poistoalgoritmi

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

Fig 8.15 [Stal05]

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

24

LRU poistoalgoritmi

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

Fig 8.15 [Stal05]

NRU (Not Recently Used) -poistoalgoritmi

- n Sivukehyksissä viitebitti (R, Reference)
 - u nollataan aika ajoin vaatii siis laitteistotukea!
 - u laitteisto asettaa, jos viittaus
- n Sivukehyksessä muutettu bitti (M, Modified)
 - u nollataan levyille kirjoituksen yhteydessä
 - u laitteisto asettaa, jos kirjoitus
 - F tarvitaan siis joka tapauksessa!
- n Poista
 - u joku, johon ei viittausta ja jota ei muutettu
 - u joku, johon ei viittausta ja jota muutettu
 - u joku, johon viittaus ja jota ei muutettu
 - u joku
- n Helppo toteuttaa, vähän HW-tukea

FIFO-poistoalgoritmi

- n Korvaa sivu, joka ollut kauimmin muistissa
- n vrt. Round-Robin vuorottamisessa
- n Helppoin toteuttaa
 - u aikaleima sivutaulun alkioon, kun tuodaan muistiin, TA1
 - u kaikki sivut listassa, muistista tuontiajan mukaisesti
- n Muistiintuontiaika korreloi huonosti käyttötärpeen kanssa
 - u kuiten muistissa ollut sivu voi olla sellainen, jota käytetään koko ajan?

jonojen hallinta?

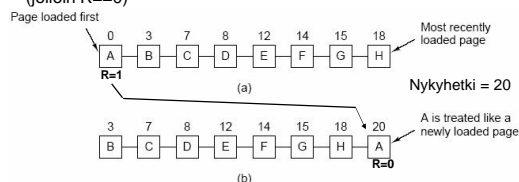
Fig 8.15 [Stal05]

RANDOM poistoalgoritmi

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

Second Chance -poistoalgoritmi

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



(Fig 4-16 [Tane01])

Clock-poistoalgoritmi

- n Second Chance -algoritmin modifikaatio
 - u ei sivujen siirtelyä listassa
- n Käy sivutilojen listaa läpi renkaana
- n Jokaiseen sivutilaan liittyy viitebitti (U eli Use bit)
 - u kun sivu muistiin, U = 0, reset silloin tällöin
 - u kun sivuun viitataan, U = 1 (MMU asettaa)
- n Korvaa sivu, jonka viitebitti on 0 Fig 8.16 [Stal05]
 - u ensimmäinen eteen sattava
- n Aseta aina tutkituissa viitebitiksi 0 Fig 8.15 [Stal05]
 - u jos arvo seuraavallakin kierroksella 0, sivuun "ei ole viitattu" (ainakaan teoriassa)
- n Aseta kaikki U=0 Fig 8.17 [Stal05]
 - u ei aina? joka 10 ms? (HW-toiminto)

Clock-poistoalgoritmi M-bitin kera

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

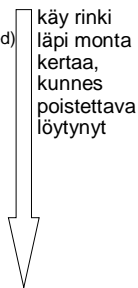


Fig 8.18 [Stal05]

Käyttöjärjestelmät II

Käyttöjoukkostrategia

Muistitilan koon hallintaan

Käyttöjoukkostrategia (Working Set Strategy)

Fig 8.1 [Stal05]

- n Sivut, joihin viitattu k:n (ikkunakoko) viimeisimmän viittauksen aikana
 - u **approksimaatio** prosessin tarvitsemille sivuille (todelliselle käyttöjoukolle)
- n Toteutus?
 - u vain approksimaatioita
 - u käyttöjoukossa sivut, joihin viitattu esim. viim. 100 ms:n aikana, kun prosessi oli suoritettavana
- n Sivut, jotka ei kuulu käyttöjoukkoon, voidaan vapauttaa
 - u pyrkii tekemään ennakoita tilaa uusille sivuille
- n Jos käyttöjoukko kutistuu pieneksi, voidaan tuoda sivuja ennalta takaisin
 - u mitkä? viitattu ja sitä seuraava?

Fig 8.19 [Stal05]

"working set" vs. "resident set"?

Käyttöjoukon koko

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

Dynaaminen käyttöjoukon koko

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

Käyttöjärjestelmät II

Levyille kirjoitus –politiikka (Cleaning Policy)

"how to clean dirty pages"

Milloin sivu kirjoitetaan levyille?

- n **Vain jos muutettu**
- n **Tarvetalletus** (demand cleaning)
 - u kirjoita levyille vasta, jos sivutila pitää ottaa käyttöön
 - u saattaa aiheuttaa pitkän viipeen sivutilan tarvisijalle
 - u entä jos kone kaatuu?
- n **Ennaltatalletus** (precleaning)
 - u kirjoitus levyille etukäteen erä kerrallaan
 - u ennaltamääritellyin aikaväleihin
- n **Sivutila siivouksen jälkeen vapaiden listaan**
 - u siivous voi olla hukka-investointi
 - u entäpä, jos sivuun viitataan taas?
- n **Entäpä jos levy odottaa toimitonna, eli sillä on pieni utilisaatio?**

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

37

Puskurointi (page buffering)

- n Pidä tietty osa sivutiloista vapaana
 - u joskus tulee vapautetuksi väärä sivu
- ~ sivujen välimuisti
 - u nopea palautus takaisin käyttöön sivupuskurista, jos "vapaassa" sivukehyksessä olevaan tietoon viitataan
- n Poistettavaksi merkitty sivu lisätään
 - u vapaiden listaan, jos ei muutettu
 - u muutettujen sivujen listaan, jos muutettu
- n Sivu pysyy silti alkuperäisellä paikalla muistissa
 - u vain merkintä prosessin sivutaulussa poistetaan
- n Varaus vapaiden listan alusta
 - u viimeksi vapautetuilla suurempi mahdollisuus tulla "pelastetuksi"
 - F miksi?

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

38

Käyttöjärjestelmät II

Moniajoaste

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

39

Moniajoaste (load control)

- n Löydettävä sopiva suorituksessa (muistissa) olevien prosessien lkm
- n Liian vähän?
 - u CPU jouten, jos prosessit Blocked tilassa Fig 8.11 [Stal05]
- n Liian paljon?
 - u ruuhkautuminen (trashing): sivuttaminen vie liikaa aikaa Fig 8.21 [Stal05]
- n Mitä tehdä jos liikaa prosesseja?
Eli siis liian vähän muistia per prosessi?
 - u heittövaihtoa Ready-prosesseja levyille
 - F se, jolla pienin prioriteetti
 - F se, joka aiheutti puutoksen
 - F se, joka oli edellisessä suorituksessa
 - F se, jolla pienin käyttöjoukko
 - F se, jolla eniten muistia käytössä tai jolla suurin aikaviipale
 - F kuka vaan, mutta joku!

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

40

Moniajoasteen dynaaminen kontrolli

- n **L=S -kriteeri**
 - u sivunpuutosten väliaika L
 - u sivunpuutosten käsittelyaika S
 - u $L \gg S \rightarrow$ kasvata mpl, $L \ll S \rightarrow$ pienennä mpl
- n **50% sääntö**
 - u Util (sivutuslevy) olisi hyvä olla 50%
 - u "pieni" U \rightarrow kasvata mpl, "iso" U \rightarrow pienennä mpl
- n **Globaali Clock poistoalgoritmi, viisarin nopeusrajat**
 - u "hidas" \rightarrow kasvata mpl
 - u "nopea" \rightarrow pienennä mpl

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

41

Käyttöjärjestelmät II

Heittövaihtoalue

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

42

Heittovaihtoalue (swap area, VM backup store)

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

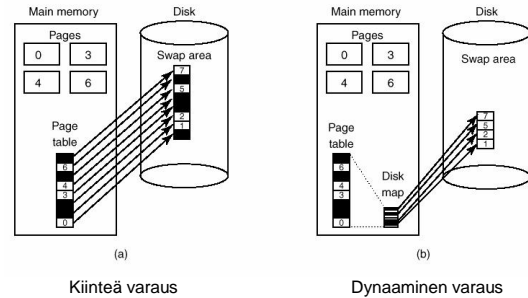
KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

43

Heittovaihtoalue

(Fig 4-33 [Tane01])



KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

44

Kertauskysymyksiä

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

KJ-II K2006 / Auvo Häkkinen - Teemu Kerola

22.3.2006

45