Protsessoriaja planeerimine

Ülevaade

- Üldised põhimõtted
- Plaanimise eesmärgid
- Plaanimisalgoritmid
- Plaanimine mitme protsessoriga süsteemides
- Plaanimine reaalajasüsteemides
- Algoritmide hindamine

Üldised põhimõtted

- Multiprogrammeerisega saab protsessori maksimaalselt ära kasutada
- Protsessori ja I/O kasutuse ”pursete” (*burst*) tsüklid — protsessi täitmisel vahelduvad protsessorikastus ja I/O
- Protsessorikasutususe faasi pikkus varieerub

Planeerija

- Lühiajaline planeerija valib valmis protsesside nimekirjast sobiva ja annab talle protsessoriaja
- Planeerimisotsuseid võidakse teha järgmistes situatsioonides:
  1. Protsess läheb tööolekust ooteolekusse
  2. Protsess läheb tööolekust valmis olekusse
  3. Protsess läheb ooteolekust valmis olekusse
  4. Protsess lõpetab töö
- Juhud 1 ja 4 on mitteväljatõrjuvad
- Muud juhud on väljatõrjuvad

Dispetšer

- Dispetšer annab juhtimise lühiajalise planeerija poolt valitud protsessile:
  - Kontekstivahetus
  - Kasutajatasemele lülitumine
  - Hüppamine sellele aadressile, kus programm jätkuma peaks
- Dispetšeri viivitus (*dispatch latency*) — kui kaua võtab dispetšeril aega eelmise programmi peatamine ja uue käivitamine

Plaanimise eesmärgid

- Protsessoriaja maksimaalne ära kasutamine — hoida protsessor võimalikult pidevalt töös
- Läbiaskevõime — protsesside arv, mis ajaühikus oma töö tehtud saab
- Valmissaamise aeg (*turnaround time*) — kui kaua aega võtab ühe protsessi valmis saamine
- Ooteaeg — kui kaua aega protsess veedab valmis järjekorras oodates
- Vastama hakkamise aeg — kui kaua võtab aega protsessi käivitamise soovist esimese väljundini jõudmiseni

Optimeerimiskriteeriumid

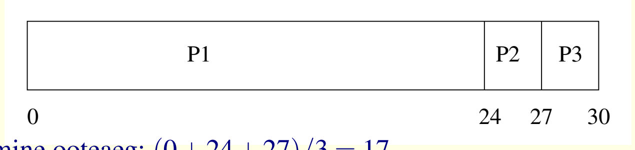
- Maksimeerida protsessoriaja kasutus
- Maksimeerida läbilaskevõime
- Minimeerida valmissaamise aeg
- Minimeerida ooteaeg
- Minimeerida vastama hakkamise aeg
- Vahel tahame ka aegade dispersiooni vähendada (et ajad ennustatavad oleksid)

FCFS (*First Come First Served*)

- Protsesside protsessorisoovid täidetakse saabumise järjekorras

Protsess	Ajahulk
P_1	24
P_2	3
P_3	3

- Gantti diagramm:



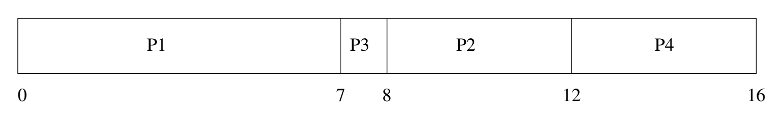
- Keskmine ooteaeg: $(0+24+27)/3 = 17$
- Saabumise järjekord P_2, P_3, P_1, siis keskmine ooteaeg 3
- Konvoiefekt

Lühim töö kõigepealt (SJF)

- SJF — *Shortest Job First*
- Seome iga protsessiga järgmise protsessorisoovi ajalise pikkuse, kasutame seda lühima järgmise soovija valimiseks
- Mitteväljatõrjuv variant — protsessoriaega ei saa vägisi ära võtta
- Väljatõrjuv variant — kui saabub väiksema protsessoriaja sooviga protsess, siis katkestatakse käesolev. Seda kutsutakse ka *Shortest-Remaining-Time-First* (SRTF)
- SJF on tõestatavalt optimaalne — minimiseerib keskmise ooteaja

Mitteväljatõrjuva SJF näide

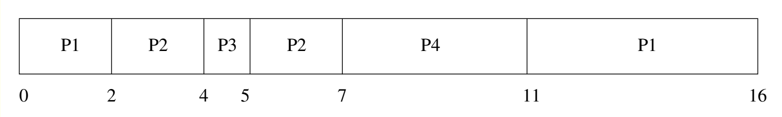
Protsess	Saabumisaeg	Ajahulk
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4
- SJF (mitteväljatõrjuv)



- Keskmine ooteaeg $(0+6+3+7)/4 = 4$

Väljatõrjuva SJF näide

Protsess	Saabumisaeg	Ajahulk
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4
- SJF (väljatõrjuv)



- Keskmine ooteaeg $(9+1+0+2)/4 = 3$

Järgmise ajakasutuse pikkuse hindamine

- Saame ainult ennustada varasema järgi
- Näiteks eksponentsiaalse keskmise kaudu:
  - t_n — n. sammu kulunud aeg
  - τ_{n+1} — n+1. sammu ennustatav aeg
  - α $(0 \leq \alpha \leq 1)$
- Defineerime

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$

Prioriteedi järgi plaanimine

- Iga protsessiga seotakse prioriteet
- Prioriteedid on staatilised või dünaamilised
- Protsessori saab suurima prioriteediga protsess (sama prioriteedi korral FCFS)
- Nii väljatõrjuv kui mitteväljatõrjuv
- SJF on prioriteete kasutav, prioriteedi ennustab järgmine protsessorikasutusaeg
- Näljutamine (*starvation*) — lahenduseks protsesside ea arvestamine

Ringiratast planeerimine

- Ingl. k *Round-Robin* (RR)
- Igale protsessile antakse väike jupp protsessoriaega (ajakvant), harilikult 10 - 100 ms
- Selle aja möödudes tõrjutakse protsess välja ja pannakse valmis-järjekorra lõppu
- Kui järjekorras on n protsessi ja ajakvandi pikkus on q, siis iga protsess saab $1/n$ osa protsessoriajast kuni q ühiku kaupa korraga. Ükski protsess ei oota kauem kui $(n-1)q$ ühikut.
- q on suur $\Rightarrow$ FIFO
- q peab olema siiski piisavalt suur kontekstivahetusega võrreldes, muidu kulub planeerimisele liiga suur osa ajast
- Veidi pikem keskmine ooteaeg kui SJF, aga parem reageerimiskiirus

Mitmetasemeline järjekord

- Ingl. k. *multilevel queue*
- Valmis-järjekord lõhutakse mitmeks eraldi järjekorraks, milledest igal võib olla oma planeerimisalgoritm. Näiteks:
  - Esiplaan (interaktiivsed tööd) — Round Robin
  - Taustatööd (pakktöötlus) — FCFS
- Kuidas erinevate järjekordade vahel aega jagada?
  - Fikseeritud prioriteet — kõigepealt kõik esiplaanil olijad, kui neid pole, siis taustatööd väljatõrjutavalt
  - Ajakvantidega — igale järjekorrale mingi protsent kogu protsessoriajast (näiteks 80% esiplaanil olijatele RR jaoks ja 20% taustatöödele FCFS jaoks)

Mitmetasemeline tagasisidega järjekord

- Protsessil lastakse mitme järjekorra vahel liikuda (paindlikkus!)
- Arvutusmahukad kolitakse automaatselt allapoole, I/O mahukad ja vanad protsessid automaatselt ülespoole
- Konkreetse mitmetasemelise tagasisidega järjekorra defineerimiseks on vaja järgmisi parameetreid:
- Järjekordade arv
- Iga järjekorra sisemised planeerimisalgoritmid
- Meetod määramaks, millal protsess kõrgema prioriteedi järjekorda ülendada
- Meetod määramaks, millal protsess madalama prioriteedi järjekorda tagandada
- Meetod määramaks, kuhu järjekorda protsessid algul satuvad

Mitmetasemeline tagasisidega järjekord: näide

- Kolm järjekorda:
  - Q_0 — ajakvant 8 millisekundit
  - Q_1 — ajakvant 16 millisekundit
  - Q_2 — FCFS
- Planeerimine
  - Uued protsessid saabuvad järjekorda Q_0 ja lisatakse sinna lõppu
  - Q_0 (ja Q_1) sees valitakse täitmiseks järjekorra esimene
  - Q_0 sees antakse aega 8 ms, kui selle jooksul tööd tehtud ei saa, pannakse Q_1 lõppu
  - Kui Q_0 on töödeldud, aetakse aega Q_1 protsessidele. Kui protsess ei saa ka 16 ms ajakvandiga hakkama, pannakse ta Q_2 lõppu
  - Q_2 sees kasutatakse FCFS (Q_0 ja Q_1 poolt üle jäävas ajal)

Planeerimine mitme protsessoriga süsteemides

- Planeerimine on keerulisem, kui masinas on mitu protsessorit
- Homogeensed protsessorid (SMP)
- Koormuse jagamine
- Üks või mitu valmis-järjekorda?
- Asümmeetrilised süsteemid — ainult üks protsessor puudutab süsteemseid struktuure, andmete jagamist on vähem vaja
- Protsesside migreerimine ühelt protsessorilt teisele

Reaalajaline planeerimine

- Range reaalaeg — kriitilise protsessi mingi lõik tuleb garanteeritult mingi aja jooksul täita
  - Ressursside reserveerimine
  - Ei salvestusseadmeid ega virtuaalmälu
- Mitterange reaalaeg — kriitilised protsessid peavad olema prioriteetsemad kui mittekriitilised
  - Protsesside ebavõrdsus, isegi näljutamine
  - Ajas mittevähenev prioriteet
  - Dispetšeri viivitus peab minimaalne olema (raske — I/O seadmete taga ootamine näiteks)
- Väljatõrjumispunktid vs iseennast väljatõrjuv tuum
- Prioriteedi inversioon ja prioriteedi pärimine

Algoritmide hindamine

- Defineerime ootused
- Analüütiline hindamine
- Deterministlik modelleerimine — võtab mingi ettantud töökoormuse ja mõõdab iga algoritmi omadused
- Järjekordade võrk
  - Little'i valem: $n = \lambda \times W$
- Simulatsioon
  - Juhuslik protsesside tekke ja tegevuse genereerimine
  - Salvestatud jäljed (*traces*) tegelikust protsesside tekkest
- Päris realiseerimine ja tegelikus töös mõõtmiste tegemine
  - Keskkond muutub (ka planeerija enda tõttu)

Lõimede planeerimine

- Protsessisisene planeerimine — kuidas kasutajataseme lõimesid protsessi sees planeerida (millis(t)ele neist anda tuuma lõim(ed))
- Ülesüsteemne planeerimine — missugustele tuuma taseme lõimedele protsessoriaega anda

Näide: Solaris

- Prioriteedi järgi planeerimine
- 4 prioriteediklassi:
  - Reaalaja
  - Süsteemi (tuuma lõimed, väljatõrjuvad + prio)
  - Ajajaotus (vaikimisi)
  - Interaktiivsed
- Igal protsessil algul 1 LWP, LWP-d pärivad prioriteedi protsessilt
- Ajajaotusklassis mitmetasemeline tagasisidega planeerimine dünaamiliste prioriteetidega
- Interaktiivne klass: ajajaotussüsteem, kus aknasüsteemi rakendustel on suurem prioriteet
- Klasside sees arvutatud prioriteetidest valitakse globaalselt kõrgeim

Näide: Windows 2000

- Väljatõrjuv prioriteetidel baseeruv ajakvantidega süsteem
- 32 prioriteeditaset, igale oma järjekord

  0 — mäluhaldus

  1-15 — harilik klass

  16-31 — reaalajaklass
- Win32 API defineerib 6 prioriteediklassi:
  - Reaalaja
  - Kõrge
  - Kõrgem kui normaalne
  - Normaalne
  - Alla normaalse
  - Idle
- Igas klassis omaette 6 suhtelise prioriteedi taset (kokku 6×6 tabel prioriteetidega)
- Esiplaani aknaga protsess saab boonust