Tosiaikajärjestelmät – Luento 9: Moniprosessorijärjestelmät

Tiina Niklander

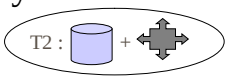
Liu: Real-Time Systems luku 9

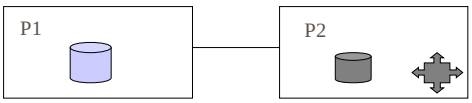
Sisältö

- Järjestelmämalli
  - moniprosessorikone
  - hajautettu järjestelmä
- Päästä-päähän
  - Tehtävän töiden jako prosessoreille
  - Resurssien jakautuminen prosessoreille
  - Prosessorien välinen kommunikointi
- Tehtävien sijoittelu prosessoreille
- Moniprosessorien prioriteettikatto (MPCP)
- Globaali synkronointi

Keskeiset kysymykset

- Töiden sijoittelu
- Resurssien hallinta ja varaus
- Prosessorien yhteinen tilatieto





T2 on suoritettavana osittain molemmilla prosessoreilla. Resurssia tarvitseva osa suoritetaan aina resurssin kanssa samassa paikassa

Järjestelmämalli

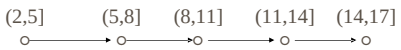
- Moniprosessorikone
  - tiukasti kytketty (tightly coupled)
  - globaali tilatieto saatavilla vähin kustannuksin
- Hajautettu järjestelmä
  - löyhästi kytketty (loosely coupled)
  - globaalin tilatiedon saanti on kallista
- Tavallinen PC on mallin kannalta moniprosessorikone
  - CPU ja levy 'erilliset prosessorit'

Järjestelmämalli

- Prosessorit
  - kullakin prosessorilla oma vuorotusmekanismi, resurssien hallinta ja synkronointi
  - Prosessorit ovat identtiset, jos ne voidaan vaihtaa keskenään
  - Kun prosessorit eivät ole identtiset, voi tehtävän suoritusaika vaihdella sijoituspaikan perusteella
- Prosessien välinen yhteistoiminta töiden vuorotuksessa edellyttää globaalia tilatietoa
  - Eri menetelmät tarvitsevat erilailla tuota tietoa
  - Menetelmien vertailu perustuu tarvittavan tiedon määrään ja menetelmän riippuvuuteen tiedon saatavuudesta

Päästä-päähän tehtävä

- Vierailee useilla prosessoreilla ja jakautuu useisiin töihin, jotka ketjutettu



- Esim. lennonvalvonnassa uuden lentokoneen saapuminen
  1) reitinlaskenta tutkatiedon perusteella (signaaliprosessori)
  2) reittitiedon tallennus dataprosessorilla
  3) reittitiedon vertaaminen muihin tallennettuihin reitteihin (dataprosessori)

Päästä-päähän tehtävien osatöiden sijoittelu prosessoreille

- Mallinnetaan tehtävät teoreettisten ja soveltuvien töiden jaottelumallien avulla
  - Yhden tehtävät peräkkäiset työt suoritetaan eri prosessoreilla (mallin oletus) – Vaihtoehtoiset mallit
    - Job shop: Töiden välissä vaihdetaan prosessoria vapaasti, järjestyksestä riippumatta
    - Flow shop: Kaikkien tehtävät noudattavat samaa prosessori järjestystä töiden edetessä
  - mallilla on käyttöä mm. sijoitettaessa ihmisiä ja työtehtäviä, tapaamisia kokoushuoneisiin, töitä prosessoreille jne.
  - Ratkaisualgoritmeja on useita erilaisia
  - Yleisessä tapauksessa ratkaisu on NP-täydellinen

Yleinen jaotteluongelman kuvaus

- **Syöte**: suunnattu syklitön verkko G=(V,E), jonka solmut kuvaavat töitä ja kaaret niiden suoritusten peräkkäisyyttä. Kaari (u,v) kuvaa, että työn u pitää valmistua ennen työtä v.
- **Ongelma**: Millainen töiden sijoittelu ja ajoitus suorittaa kaikki kuvatut tehtävät minimoiden (optimoiden) joko prosessorien määrän tai käytettävän ajan?
- Töiden kulku
  - palvelupisteellä on rajoitettu kapasiteetti
  - Työt suoritetaan yksi kerrallaan, seuraava saa aloittaa vasta edellisen valmistuttua

Job shop – yleinen sijoittelumalli

- Parametrit:
  - i – tehtävät (tasks)
  - j – työt (jobs), tehtävän sisäisiä
  - k – prosessorit
  - ($e_{i,j,k}$ – työn suorituksen kesto)
- Esimerkki:

	Työ 1	Työ 2	Työ 3
Tehtävä 1	5	6	-
Tehtävä 2	2	7	2

- Tehtävällä 1 on kaksi työtä, jotka suoritetaan prosessoreilla 5 ja 6. Tehtävällä 2 on kolme työtä, jotka suoritetaan vastaavasti prosessoreilla 2,7 ja 2.

Flow shop - malli

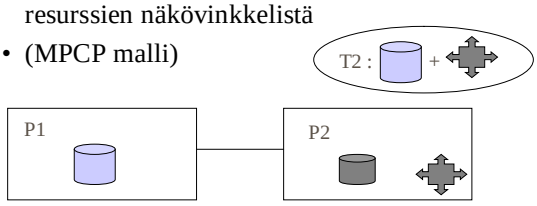
- Flow shop on job shop -mallia rajoitetumpi, koska
  - Kaikkien tehtävien töiden järjestys noudattaa samaa prosessorisekvenssiä.
  - Töiden suoritusajat voivat vaihdella, mutta prosessorien käyttöjärjestys ei.
  - Jos jollakin tehtävällä ei ole työtä tietylle prosessorille, se mallinnetaan nollan mittaisena suoritusaikana.
- Edeltäjyysvaatimus on vastaava kuin job shop – mallissa:
  - työ i-1 prosessorilla i-1 on valmis ennen kuin työ i prosessorilla i voi alkaa.

Tosiaikaisuuden lisäykset

- Tehtävillä on aikarajoja
  - Globaali aloitusaika (end-to-end release time) on ensimmäisen työn aloitushetki
  - Globaali takaraja (end-to-end deadline) on viimeisen työn takaraja
  - Väliajoilla ei ole sinänsä merkitystä onnistumiselle
- Tehtävät voivat olla jaksollisia

Vaihtoehtoinen mallinnustapa

- Tarkastellaan tilanne ei tehtävien vaan resurssien näkövinkkelistä
- (MPCP malli)



T2 voidaan suorittaa yhtenä tehtävänä, joka tarvitsee myös etäresursseja (ns. globaali kriittinen osio)

Mallien erot:

- Päästä-päähän malli:
  - Yksi työ vain yhdellä prosessorilla
- MPCP malli:
  - Työ voi käyttää myös globaaleja resursseja ns. globaalin kriittisen osion kautta.
- **HUOM**: työ ei saa kriittisen osionsa aikana käyttää kuin yhden prosessorin lokaaleja resursseja

Sisältö

- Järjestelmämalli
  - moniprosessorikone
  - hajautettu järjestelmä
- Päästä-päähän
  - Tehtävän töiden jako prosessoreille
  - Resurssien jakautuminen prosessoreille
  - Prosessorien välinen kommunikointi
- Tehtävien sijoittelu prosessoreille
- Moniprosessorinen prioriteettikatto (MPCP)
- Globaali synkronointi

Tehtävien sijoittelu

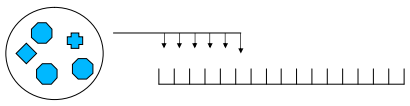
- Pelkästään suoritusaikojen perusteella
  - Lokerointiongelma (Bin-packing)
  - First-Fit (tai RMFF, Rate-Monotonic-First-Fit)
  - RMST (Rate-Monotonic Small Tasks)
- Huomioidaan lisäksi kommunikointiviipeet
  - Integer Linear Programming
- Näiden lisäksi mukaan resurssikilpailu
  - Graph Partitioning

Lokerointiongelma

- Miten sijoitellaan saadut palikat mahdollisimman taloudellisesti annettuihin lokeroihin?
- Lokeron koko
  - kiinteä
  - vaihteleva, riippuu esim. sijoitettujen lukumäärästä
- Minimoidaan tarvittavien lokeroiden määrä!
- Optimaalinen ratkaisu on yleisessä tapauksessa NP-täydellinen
- Hyviä heuristisia menetelmiä olemassa (mm. First-Fit)

First-Fit

- Lokerointiongelman heuristinen ratkaisu
- Sijoitellaan palikat lokeroihin satunnaisessa järjestyksessä
- Kukin palikka laitetaan ensimmäisen lokeroon, johon se vielä sopii
- Jos palikka ei sovi jo käytössä oleviin lokeroihin, otetaan uusi lokero



Tehtävien sijoittelu prosessoreille

- MPCP-malli
- Lokeron koko on käyttöasteen maksimi kuten ajoitettavuusanalyysissä
- Sijoiteltavien palikoiden koko on kunkin tehtävän tarvitsema käyttöaste
- Jos prosessorien määrä kiinteä *m*, niin First-fit löytää aina ajoituksen vain kun sijoiteltavien töiden yhteenlaskettu käyttöaste on korkeintaan

$$U_{FF} = m(2^{1/2} - 1) = 0.414m$$

RMFF algoritmi

- First Fit muunnelma
- Rate-monotonic ajoitus prosessoreilla
  - Käytetään lokeron koon ylärajana RM:n tarkkaa ylärajaa

$$u_i + U_k \le n(2^{1/n} - 1)$$

- Järjestetään työt jakson pituuksien perusteella:
  - Lyhimmät ensin
  - Tämä on linjassa RM:n priorisoinnin mukaan

RMFF

Ti	(pi,ei)	ui
T1	(2,1)	0.500
T2	(2.5,0.1)	0.040
T3	(3,1)	0.333
T4	(4,1)	0.250
T5	(4.5,0.1)	0.022
T6	(5,1)	0.200
T7	(6,1)	0.167
T8	(7,1)	0.143
T9	(8,1)	0.125
T10	(8.5,0.1)	0.012
T11	(9,1)	0.111

Sijoittelujärjestys on numerojärjestys

P1 T1, T2, ~~T3~~

U = 0.500 < 1.000
U = 0.540 < 0.828
U = 0.873 > 0.780

$U_{RM} \le n(2^{1/n} - 1)$

≈ 0.693

määrä	U_{RM}
1	1.000
2	0.828
3	0.780
4	0.757
5	0.743
6	0.735
7	0.729

RMFF

Ti	(pi,ei)	ui
T1	(2,1)	0.500
T2	(2.5,0.1)	0.040
T3	(3,1)	0.333
T4	(4,1)	0.250
T5	(4.5,0.1)	0.022
T6	(5,1)	0.200
T7	(6,1)	0.167
T8	(7,1)	0.143
T9	(8,1)	0.125
T10	(8.5,0.1)	0.012
T11	(9,1)	0.111

Jatketaan tehtävästä T3

P1 T1 T2 ~~T3~~ T4 T5 ~~T6~~

U = 0.540
U = 0.790 > 0.780
U = 0.562 < 0.780
U = 0.762 > 0.757

P2 T3 T4 ~~T5~~ T6

U = 0.333 < 1.000
U = 0.583 < 0.828
U = 0.783 > 0.780

RMFF

Ti	(pi,ei)	ui
T1	(2,1)	0.500
T2	(2.5,0.1)	0.040
T3	(3,1)	0.333
T4	(4,1)	0.250
T5	(4.5,0.1)	0.022
T6	(5,1)	0.200
T7	(6,1)	0.167
T8	(7,1)	0.143
T9	(8,1)	0.125
T10	(8.5,0.1)	0.012
T11	(9,1)	0.111

Jatketaan tehtävästä T7

P1 T1 T2 T5 T7 T10

U = 0.562
U = 0.729 < 0.757
U = 0.741 < 0.743

P2 T3 T4 T8

U = 0.583
U = 0.726 < 0.780

P3 T6 T9 T11

U = 0.200 < 1.000
U = 0.325 < 0.828
U = 0.436 < 0.780

RMST algoritmi

- Rate-Monotonic Small Tasks (RMST)
  - Hyödyntää RM:n käyttäytymistä harmoonisten jaksojen kanssa.
  - Kun tehtävien jaksojen pituudet lyhimmän monikertoja, niin käyttöasteen maksimi kasvaa merkittävästi (jopa ykköseen)
- Työt sijoitellaan jaksonpituudesta lasketun arvon X_i määräämässä suuruusjärjestyksessä

$$X_i = \log_2 p_i - \lfloor \log_2 p_i \rfloor$$

RMST algoritmi

- Ajoitettavuusehto prosessorille on

$$u_i + U_k \le \max(\ln 2, 1 - \zeta_k \ln 2), \text{ missä } \zeta_k = \max X_l - \min X_l$$

- Työt ovat sijoiteltavissa m:lle prosessorille, jos työkuormasta laskettu käyttöaste on pienempi kuin

$$U_{RMST} = (m-2)(1-u_{\max}) + 1 - \ln 2, \text{ kun } m > 2$$

missä u_{max} on töiden suurin yksittäinen käyttöaste

RMST

Ti	(pi,ei)	ui	Xi
T1	(2,1)	0.500	0
T2	(2.5,0.1)	0.040	0.322
T3	(3,1)	0.333	0.585
T4	(4,1)	0.250	0
T5	(4.5,0.1)	0.022	0.170
T6	(5,1)	0.200	0.322
T7	(6,1)	0.167	0.585
T8	(7,1)	0.143	0.807
T9	(8,1)	0.125	0
T10	(8.5,0.1)	0.012	0.087
T11	(9,1)	0.111	0.170

Ti	(pi,ei)	ui	Xi
T1	(2,1)	0.500	0
T4	(4,1)	0.250	0
T9	(8,1)	0.125	0
T10	(8.5,0.1)	0.012	0.087
T5	(4.5,0.1)	0.022	0.170
T11	(9,1)	0.111	0.170
T2	(2.5,0.1)	0.040	0.322
T6	(5,1)	0.200	0.322
T3	(3,1)	0.333	0.585
T7	(6,1)	0.167	0.585
T8	(7,1)	0.143	0.807

RMST

Ti	(pi,ei)	ui	Xi
T1	(2,1)	0.500	0
T4	(4,1)	0.250	0
T9	(8,1)	0.125	0
T10	(8.5,0.1)	0.012	0.087
T5	(4.5,0.1)	0.022	0.170
T11	(9,1)	0.111	0.170
T2	(2.5,0.1)	0.040	0.322
T6	(5,1)	0.200	0.322
T3	(3,1)	0.333	0.585
T7	(6,1)	0.167	0.585
T8	(7,1)	0.143	0.807

P1 T1
T4
T9
T10
T5

U = 0.500 < 1
U = 0.750 < 1
U = 0.875 < 1
U = 0.887 < 0.940
U = 0.909 > 0.882

RMST

Ti	(pi,ei)	ui	Xi
T1	(2,1)	0.500	0
T4	(4,1)	0.250	0
T9	(8,1)	0.125	0
T10	(8.5,0.1)	0.012	0.087
T5	(4.5,0.1)	0.022	0.170
T11	(9,1)	0.111	0.170
T2	(2.5,0.1)	0.040	0.322
T6	(5,1)	0.200	0.322
T3	(3,1)	0.333	0.585
T7	(6,1)	0.167	0.585
T8	(7,1)	0.143	0.807

P1 T1 T4 T9 T10 — U = 0.887, U = 0.750 < 1

P2 T5 T11 T2 T6 T3 — U = 0.022 < 1, U = 0.133 < 1, U = 0.173 < 0.895, U = 0.373 < 0.895, U = 0.706 < 0.712

P3 T7 T8 — U = 0.167 < 1, U = 0.310 < 0.846

Esimerkkien vertailu

RMFF:

P1 (U= 0.741):
T1, T2, T5, T7, T10

P2 (U=0.583):
T3, T4, T8

P3 (U=0.436):
T6, T9, T10

RMST:

P1 (U= 0.887):
T1, T4, T9, T10

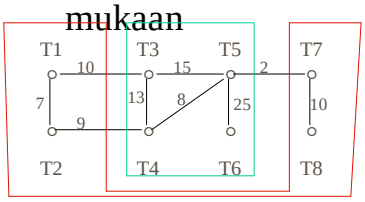
P2 (U=0.706):
T2, T3, T5, T6, T11

P3 (U=0.310):
T7, T8

Kommunikointiviipeet mukaan

- Edellä tehtävien sijoitteluun prosessoreilla vaikutti vain tehtävien suoritusaika
- Kommunikointi prosessorin sisällä on merkittävästi ’halvempaa’ kuin prosessorien välillä, joten tehtävät kannattaa sijoittaa siten, että prosessorien välinen kommunikointikustannus minimoidaan.
- Tavoitteena siis osittaa tehtävien verkko!

Kommunikointiviipeet mukaan



- Verkon ositukseen (tai oikeammin muodostettavien osien välisten kustannusten minimointiin) voidaan käyttää esim. Integer Linear Programming –menetelmää.

Ositus kaavoina

- Syötteet: i,j = 1,...,n
  - Tehtävän T_i käyttöaste u_i
  - *k:*nnen osion (prosessorin) maksikäyttöaste U_k
  - Tehtävien T_i ja T_j väliset kustannukset
    - $C_{i,j}$ – kommunikointikustannus jos eri osioihin
    - $\iota_{i,j}$ – interferenssin kustannus, jos samaan osioon
- Muodostetaan 0/1 –matriisi A, siten että
  - $A_{i,k}$ on 1, jos T_i on sijoittu k:nteen osioon, muuten arvo on 0 (i=1,.., n ja k=1,...,m)

Osituksen kaavat – jatkuu

- Rajoitteet (arvoille matriisissa):
  - Sallitut arvot 0 ja 1 $A_{i,k} = 0,1 \; \forall i = 1,..,n; k = 1,..,m$
  - Tehtävä vain yhdessä osiossa $\sum_{k=1}^{m} A_{i,k} = 1, \forall i = 1,..,n$
  - Osion käyttöaste sallituissa rajoissa

$$\sum_{i=1}^{n} u_i A_{i,k} \le U_k, \forall k = 1,..,m$$

- Minimoitava kustannusfunktio

$$C_{total} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{m}(1-\delta_{i,j})A_{i,k}A_{j,l}\left[C_{i,j}(1-\delta_{k,l}) + \iota_{i,j}\delta_{k,l}\right],$$

missä $\delta_{a,b} = 1$, kun $a = b$ ja 0, kun $a \neq b$

Sisältö

- Järjestelmämalli
  - moniprosessorikone
  - hajautettu järjestelmä
- Päästä-päähän
  - Tehtävän töiden jako prosessoreille
  - Resurssien jakautuminen prosessoreille
  - Prosessorien välinen kommunikointi
- Tehtävien sijoittelu prosessoreille
- Moniprosessorinen prioriteettikatto (MPCP)
- Globaali synkronointi

Paikallinen vuorotus

- Paikallisesti prosessori voi huomioida globaalien (täällä sijaitsevien) resurssien käyttäjät
- Multiprosessor Priority-Ceiling Protocol (MPCP)
  - Globaalien resurssien käyttäjät (työt) sijoitetaan omaan prioriteettialueeseensa paikallisesti suoritettavien töiden yläpuolelle

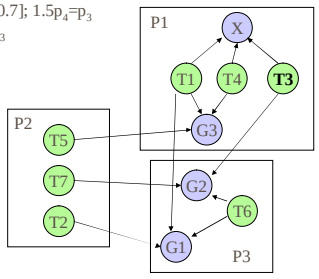
MPCP – moniprosessorien prioriteettikatto – menetelmä

- Resurssikilpailun aiheuttama estymisaika
  - Paikallinen odotus (local blocking time)
  - Paikallinen irroitusviive (local preemption delay) – syynä muualta tullut varaaja paikalliselle resurssille, korkeampi prio
  - Etäresurssin odotus (remote blocking time)
  - Etäirroituksen viive (remote preemption time) – syynä toinen muualta tullut varaaja, jolla korkeampi prioriteetti kuin itsellä
  - Viivästetty odotus (deferred blocking time) – syynä paikallisen korkeampiprioriteettisen kokema odotus

MPCP -esimerkki

T1: [X;0.5][G1;2.0][G3;0.4]; $e_{1,L} = 1.0$
T2: [G1;0.1][G1;0.5]; $14p_2 = p_3$
T3: [X;2][G2;1.0][X;0.1][G2;1.0][X;0.1][X;0.1]
T4: [X;1.0][X;2.0][X;8.0][G3;0.7]; $1.5p_4 = p_3$
T5:[G3;0.6][G3;1.0]; $p_5 = 1.5p_3$
T6:[G1;5.0][G2;0.3]
T7:[G2;1.0]

T3:n kokonaisviive
a) Paik. od. 8 (T4)
b) Paik. irr. 5.3 (T4&T5)
c) Etä od. 2*5 (T6&T7)
d) Etä irr. 9 (T2)
e) Viiv. od. 1 (T1)
Yht : 49.3
(Kaavat kirjassa s. 354-356)



Globaali synkronointi

- Miten varmistutaan töiden ja tehtävien oikeasta ajoittamisesta yli prosessorien?
  - Päästä- päähän mallissa seuraava työ saa alkaa vasta edellisen valmistuttua
  - Vastaavia rajoituksia voi olla tehtävien välillä myös MPCP-mallissa
- Prosessori saa sisäisesti vuorottaa tehtävät haluamallaan tavalla

Globaali synkronointi

- Ahne synkronoija (greedy synchronization)
  - Synkronointisignaali edellisen päätyttyä
- Vaiheen muokkaus (Phase-Modification)
  - Viivästetään seuraavaa edellisen maksimiajalla
- Muunnettu vaiheen muokkaus
  - Viivästetty aloitusaika ja synkronointisignaali
  - Myöhemmin saapunut määrä aloitushetken
- Aloituksen vartiointi (Release Guard)
  - Synkronointisignaali, mahdollinen aloitusaika jakson pituuden päässä edellisestä tai kun prosessori tulee vapaaksi (vrt. sporadinen palvelin)

Prioriteetit osatöille ?

- Päästä-päähän tehtävillä on aloitus- ja lopetusajat sekä prioriteetit
- Miten nämä ajat ja prioriteetit jaetaan osatehtäville?
- Miten jaetaan tehtävään T_i sisältyvä jousto osatöille $T_{i,k}$?
- Deadline-assignment Algorithms

Deadline assignment - vaihtoehtoja

- Ultimate Deadline (UD)

$$UD_{i,k} = D_i$$

- Effective Deadline (ED)

$$ED_{i,k} = D_i - \sum_{l=k+1}^{n(i)} e_{i,l}$$

- Proportional Deadline (PD)

$$PD_{i,k} = D_i e_{i,k} / e_i$$

- Normalized Proportional Deadline (NPD)

$$NPD_{i,k} = D_i \frac{e_{i,k} U(V_{i,k})}{\sum_{l=1}^{n(i)} e_{i,l} U(V_{i,l})}$$

Yhteenvetona

- Valittava malli vaikuttaa käytettäviin menetelmiin ja lopulta myös toteutetun järjestelmän suorituskykyyn
- Päästä-päähän mallinnus sopii hyvin tilanteeseen, jossa vähän mutta pitkiä kriittisiä alueita
- MPCP (kriittinen suoritus resurssin prosessorilla) sopii tilanteeseen, jossa paljon hyvin lyhyitä kriittisiä alueita