Kap. 10: Parallelle programmerings-modeller

- **Pensum**
  - Kap. 10.1, (10.2)
    - » turbulent og umodent fagområde
- **Programmeringsmodell**
  - en samling av abstraksjoner som gir programmereren et forenklet bilde av et underliggende datamaskinsystem
- **Kommunikasjon via delte variable**
  - felles lager, logisk sett
    - » samme adresserom
    - » (kan være felles eller distribuert lager, fysisk sett)
  - tett koplet multiprosessor
    - » (tradisjonell terminologi)
  - data må beskyttes
    - » kritiske regioner
- **Kommunikasjon via meldingsutveksling**
  - normalt vesentlig langsommere kommunikasjon
  - løst koplet multiprosessor
    - » (tradisjonell terminologi)

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Kommunikasjon via delte variable

- **Beskyttet aksess**
  - Kan realiseres på ulike måter
    - » Monitor-begrepet (OS-faget)
    - » Kritisk region
- **Kritiske regioner**
  - gjensidig utelukkelse (Eng. mutual exclusion)
    - » maks en av gangen
  - kan ikke avbrytes
  - unngå vranglås
  - rettferdighet m.h.p venting
  - realisering
    - » Test&Set (HW)
    - » Skru av/på avbrudd (på uni-prosessor)
    - » Semaforer
- **Størrelsen av en kritisk region påvirker ytelse**
  - Stor kritisk region
    - » reduserer parallellitet
  - Liten kritisk region
    - » kan bli relativt sett stor overhead

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Meldingsutveksling

- **Synkron meldingsutveksling**
  - sender og mottaker må synkroniseres
  - må vente til det finnes en komplett kommunikasjonsvei mellom de to
  - kan få blokkering, bruker ikke buffere
  - analogt med telefonsamtale
- **Asynkron meldingsutveksling**
  - synkroniserer ikke sender og mottaker
  - melding sendes som en pakke
    - » evt. venting i buffere underveis
  - vilkårlig lang tid til kvitteringssignal mottas
  - endelige buffere gjør blokkering mulig også her

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Kjøretidsplanlegging (Eng. scheduling) på multiprosessorer

- **Prosess-håndtering**
  - ulike typer prosesser
  - dynamisk oppretting, flytting (migrering), endring av tilstand
  - avveininger mellom kornstørrelse, plass-forbruk, og overhead for kommunikasjon og scheduling
- **Generelt vanskeligere enn på en uni-prosessor**
  - fordel prosessene i tid og rom slik at en totalt sett får best mulig utnyttelse av prosessorene
  - må også ta hensyn til forsinkelse ved IPC og kommunikasjonsmediets kapasitet
- **Statisk** (ved kompilering)
  - enkelt og billig
  - kan ikke ta hensyn til kjøretids-forhold
- **Dynamisk** (ved kjøring)
  - krever mer avansert OS (eller RTS)
    - » f.eks. prosess-migrering
  - automatisk lastbalansering
- **Time-sharing & space-sharing**
- **"Gang scheduling"**
  - oppstarting av en gruppe prosesser samtidig

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Eks. 10.1: Domene dekomposisjon

- **Tilnærmet beregning av integral som sum av rektangler**
- **Fig. 10.2, program nederst side 553**
  - vertsmaskin og beregningsnoder
  - numnodes( ), og mynode( )
  - hver node utfører en egen kopi av node-programmet
  - gop('+', pi, host)
    - » global operasjon på verdien 'pi', her summasjon, resultat sendes til host
    - » utføres distribuert av nodene, ordnet som et binær-tre
    - » Tid O(log *n*) der *n* er antall noder

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Data parallell model/programmering

- **Data fordeles ut over prosessorene**
  - opererer på data i parallell
  - valg av og implementasjon av datastrukturer er sentralt
    - » jevn arbeidsfordeling
    - » redusere IPC
    - » mest mulig regulær IPC
  - synkronisering skjer i størst mulig grad ved kompileringstid
    - » ved kjøretid, "i nødsfall"
  - kan realiseres
    - » SIMD
    - » SPMD (Single Program MD)
      - MIMD med synkroniseringsmekanisme
- **SIMD programmering er lettere**
  - taktvis synkron oppførsel
  - en, evt. bred, kontroll-flyt
    - » Erfaringer helt fra Illiac-IV; Oppdeling av større problemer i 64-elementers segmenter. Dagens maskiner med større antall PE'er er bedre egnet

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Data parallelle språk

- **Flere språk som er mer eller mindre omfattende modifiseringer av serielle standard-språk**
  - FORTRAN-90
  - en rekke FORTRAN-varianter
  - C* (Thinking Machines)
  - Parallel Modula 2*, 3* (Univ. i Karlsruhe)
  - m.m.fl.
- **Kompilatorer**
  - optimalisere plassering av data
  - minimalisere kommunikasjon
  - skjule maskinens fysiske dimensjoner
    - » # noder og sammenkopling
- **Portabilitet**

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Objekt-orientert parallell-programmering

- **Objekter**
  - på ulike nivåer
  - sammensatte
  - opprettes og endrer seg dynamisk
- **Objektorientering (OO) har en rekke fordeler uavhengig av seriell/parallell utførelse**
  - abstraksjon, modularitet, gjenbruk
- **OO er godt egnet for asynkron parallell utførelse**
  - COOP (Concurrent Object Oriented Programming)
  - Utveksling av meldinger mellom objekter
  - En rekke eks. finnes
    - » Actor-modell (MIT)
- **Tre typiske hovedtyper av parallellitet i objekt-orienterte systemer**
  - samlebånd, Fig. 10.3 a)
  - splitt og hersk, Fig. 10.3 b)
  - samarbeid

Copyright © 1995 Lasse Natvig, IDT/NTH/UNIT

Funksjonell programmering og logikk programmering

- **Funksjonelle programmeringsspråk**
  - Eksempler på språk:
    - » Pure LISP
      - MultiLISP
    - » SISAL
    - » Strand 88
    - » m.m.fl.
  - Eliminering av side-effekter fjerner de fleste data-avhengigheter
    - » økte muligheter for parallellitet
  - dataflyt-maskiner
- **Språk for logikk-programmering**
  - Eksempler på språk:
    - » Concurrent Prolog
    - » Parlog
    - » m.m.fl.
  - Basert på stor grad av søking, "unifisering"
    - » Unifisering er i seg selv et "iboende serielt problem"
  - Japans FGCS (Fifth Generation ComputingSystem) prosjekt
    - » PIM (Parallel Inference Machine)
    - » vellykket ??

Programvare og ytelse

- **Pensum:**
  - 10.2.1 (kursorisk)
  - forelesning-foiler
- **Ytelse**
  - partisjonering
    - » av data og program
  - parallellisering
  - avbildning av logisk prosess-struktur på fysisk prosessor-struktur
    - » statisk ("compile time")
    - » dynamisk ("run time")
  - helautomatisk
  - halvautomatisk
    - » kompilator-direktiver
  - interaktivt
    - » statisk & dynamisk analyse
    - » grafiske framstillinger
    - » hint og hjelp til programmerer om anbefalt restrukturering (ekspert-system)

Programvare og tilgjengelighet

- **Tilgjengelighet**
  - brukervennlighet
  - skalerbarhet
    - » program kan kjøre på hele maskinen og kommende utvidelser
    - » uavhengig av maskinvare
      - f.eks. høynivå-språk
      - FORTRAN-90 (Sun, IBM, TMC maskiner)
  - kompatibilitet
    - » med velkjent sekvensielt språk
    - » f.eks. fordelaktig å kunne bruke standard-biblioteker
  - portabilitet
    - » kan realiseres på ulike maskin-arkitekturer

Språk-konstruksjoner i parallelle språk, eksempler

- **FORTRAN 90 array-notasjon**
  - nedre grense : øvre grense : inkrement (stride)
  - B (1:4:3, 6:8:2, 3) angir B(1,6,3), B(4,6,3), B(1,8,3) og B(4,8,3)
    - » siste argument mangler betyr inkrement = 1
    - » * angir alle elementer
  - ved tilordning må antall elementer og formen på utvalget være kompatibelt med venstresiden
- **Et hav av ulike parallelle løkke-konstruksjoner**
  - DOALL, ENDALL
  - DOACROSS, ENDACROSS
  - ....
  - FOR ALL,
  - Pardo, ParEnd
  - FOR EACH PROCESSOR IN ProcSet
  - **Cobegin , Coend**
  - FORK - JOIN

Kap. 11: Synkronisering, prinsipper

- **Pensum**
  - Kap. 11.2.1
  - Vi var også innom synkronisering i.f.m. kap. 7.
- **Mekanismer for synkronisering og kommunikasjon**
  - Delte variable, lås, kritisk region
  - Semafor, monitor, "mailbox" m.m.
  - Meldingsutveksling
  - RPC = Remote Procedure Call
  - Client Server
  - Barriere Synkronisering
- **Realisering**
  - "read-modify-write" operasjon
  - atomisk operasjon
    - » "hard atom"
    - » "soft atom"
- **Aktiv polling (Eng. Busy-wait)**
- **Passiv venting**
  - kø av ventende prosesser
  - unngår system-belastende polling
  - økt overhead

Tre typer synkronisering

- **Lås synkronisering (pre-synkronisering)**
  - aksess til beskyttede data krever aksess til en felles nøkkel før en aksesserer data
- **Optimistisk synkronisering (post-synkronisering)**
  - 1) leser delte data
  - 2) beregner oppdatering av disse, lagres lokalt
  - 3) sjekker om noen andre har modifisert eller fått eksklusiv aksess til delte data i mellomtiden, hvis ikke får en eksklusiv aksess og oppdaterer data
  - 4) hvis noen andre har modifisert eller fått eksklusiv aksess avbrytes egen oppdatering og en starter forfra igjen på ny verdi av delte data
    - » Fordel: tidkrevende beregning av ny verdi kan holdes utenfor kritisk region
- **Server synkronisering**
  - en unik server som håndterer (og serialiserer) alle oppdateringer (Tilsv. monitor-begrepet i OS)

Kap. 12: Operativsystem for parallelle datamaskiner

- **Pensum**
  - Kap. 12.2.1, 12.2.2
- **En enkel fast master-kjerne**
  - slave-prosessorer
  - enkel, billig
  - blir lett flaskehals
  - dårlig feiltoleranse
- **Flytende kjerne**
  - Tre varianter, Fig. 12.4