Kap. 7: Multiprosessorer og multimaskiner

- **Sammenkopling i multiprosessorer (Eng. MP system interconnect)**
  - hierarkiske buss-systemer
  - crossbar-svitsj
  - flerports-lager
  - flertrinnsnettverk
  - kombinerende nettverk
- **Hurtigbuffer-koherens problemet**
  - forklaring av problemet
  - løsningsteknikker
    - » buss-snusing
    - » katalogbaserte protokoller
- **Litt om synkronisering**
- **Multimaskiner**
  - tre generasjoner
- **Meldingsutvekslings-mekanismer**

Sammenkoplingsnettverk i multiprosessorer

- **Fig. 7.1**
  - ulike nettverk for ulike behov
  - ulike realiseringer
- **topologi**
- **"timing control"**
  - synkron
  - asynkron
- **svitsje-metode**
  - linjesvitsjing
  - pakkesvitsjing
- **nettverks-kontroll**
  - sentralisert
  - desentralisert

(Hierarkiske) buss-systemer

- **Ulike typer busser på ulike nivåer**
  - Fig. 7.2
  - Bakgrunns-stoff som forutsettes kjent
- **I datamaskinarkitektur fokuserer vi på**
  - logiske aspekter ved busser
  - anvendelser av busser som (sammensatt) komponent i konstruksjon av et system
- **Hierarkiske busser og hurtigbuffere**
  - Fig. 7.3 (Tekst side 334 **og 357!**)
    - » klynger av prossesorer
    - » en klynge tilsv. en buss-basert multiprosessor
    - » sammenkopling av klynger

- **Pensum:**
  - tekst om Futurebus+ på side 335, fig. side337, 355-356, og 364 utgår
  - "wired barrier..." side 365--367 utgår

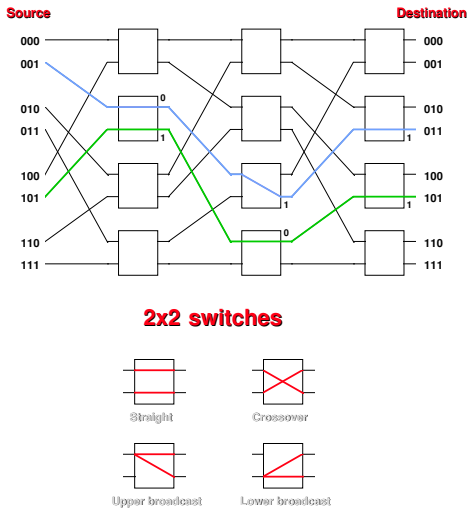
Krysskoplingsnett og flerport-lager

- **Generelt for svitsj-baserte nettverk**
  - dynamiske forbindelser
  - ett-trinns nettverk
    - » resirkulerende
    - » billigere
    - » hva er forskjellen på *n* iterasjoner i et ett-trinns-nettverk og en iterasjon i et *n*-trinns nettverk?
- **Krysskoplingsnett (Eng. crossbar-switch)**
  - Fig. 2.26, 7.6
  - små og middelsstore systemer
  - Kostnad $O(n^2)$
  - forskning på optiske løsninger
- **Flerport-lager**
  - Fig. 7.7

Flertrinnsnettverk

- **ikke-blokkerende**
  - kan utføre enhver permutasjon av innganger til utganger i en iterasjon
- **blokkerende**
  - Eks. Omega-nettverk, (repetert på neste foil)
  - noen permutasjoner kan kreve flere iterasjoner
  - fig. 7.8, 7.9
  - "Butterfly"-nettverk
    - » en spesialtype av Omega-nettverk uten kringkasting
- **kombinerende nettverk (Eng. "combining nettwork")**
  - "hot spot" problemet
    - » overbelastet del av lagersystemet
    - » lagerlokasjon som aksesseres av (for) mange
    - » f.eks. synkroniseringsvariable
  - Fetch&Add(*x*,*e*)
    - » en atomisk lager-operasjon som kan realiseres i ett flertrinnsnettverk for å redusere hot-spot-probl.
    - » returner verdi av *x*, legg til *e*
    - » Fetch&Add(*X*,*1*) utført av *N* prosessorer samtidig gir en serialisering av prosessorene der hver får returnert en verdi mellom 1 og *N*, hvis *X* initielt = 0
    - » Fig. 7.11

Ruting i Omega-nettverk



Hurtigbuffer koherens

- **Motivasjon for private hurtigbuffere**
  - Multiprosessorer med delt lager, 3 hoved-problemer
    - » lager-kollisjon (Eng. memory contention)
    - » "svitsj"-kollisjon (Eng. communication contention)
    - » aksess-tid til lager
- **Konsistens = koherens**
- **Inkonsistens**
  - ulike kopier av samme lagerlokasjon har ulik verdi
    - » innen samme nivå
    - » mellom to ulike nivåer
- **Repetisjon**
  - Write Through (WT) = "tversigjennomskriving"
  - Write Back (WB) = utsatt tilbakeskriving
- **<u>Mulige</u> årsaker til inkonsistens**
  - delte skrivbare data, Fig. 7.12 a)
  - prosess migrering, Fig. 7.12 b)
  - I/O aktivitet, Fig. 7.13

Løsninger av hurtigbuffer koherens problemet

- **Statiske løsninger**
  - programvare baserte metoder
  - kompilator bidrar vesentlig
- **Dynamiske løsninger**
  - maskinvarebasert metode
  - Kombinasjoner mellom SW/HW
- **Hurtigbuffer koherens-taktikker (Eng. policy)**
  - Fig. 7.14
  - **skriv-ugyldiggjør** (Eng. write-invalidate)
  - **skriv-oppdater** (Eng. write-update)
- **Konsistens kommandoer**
  - navn på datatrafikk for å opprettholde konsistens

Statiske løsninger på hurtigbuffer-koherens problemet

Statiske løsninger:

1) Delte hurtigbuffre
ikke benytte private hurtigbuffre

2) Buffre "ufarlige" data
ikke delte skrivbare data

3) Buffre "ufarlige" og beskyttede data
Alle delte skrivbare data beskyttes med låser (kritiske regioner), slik at disse data (untatt selve låsen) blir ufarlige. Krever eksplisitt tømming av hurtigbuffer ("Cache flush")

Metode 2) og 3) er programvarebaserte

Dynamiske løsninger på hurtigbuffer-koherens problemet

Dynamiske løsninger

4) Sentralisert global tabell (eks. på katalog-metoden)
- katalog: hvilke hurtigbuffre har kopier av en blokk
- lagrer tilstand til hver blokk
- kostbar løsning, vanligvis realisert som del av hovedlager-kontroller
- hurtigbuffer-konsistenskommandoer kan sendes selektivt til bare de aktuelle prosessorene ("multicast")

5) Distribuert tabell, buss-snusing (Eng. bus-snooping)
- Tilstand til hver hurtigbuffer-blokk lagres distribuert sammen med hurtigbufferet
- Prosessoren har en buss-kontroller som lytter kontinuerlig på bussen (snuser, spionerer)
- relativt enkelt og billig, mange varianter er realisert
  - » Goodman's Write-Once, DEC's Firefly, Synapse, Berkeley, Dragon, Illinois
- konsistenskommandoer kringkastes

- **Metode 4) og 5) er maskinvarebaserte**

Tilstandsdiagram for skriv-ugyldiggjør protokoll

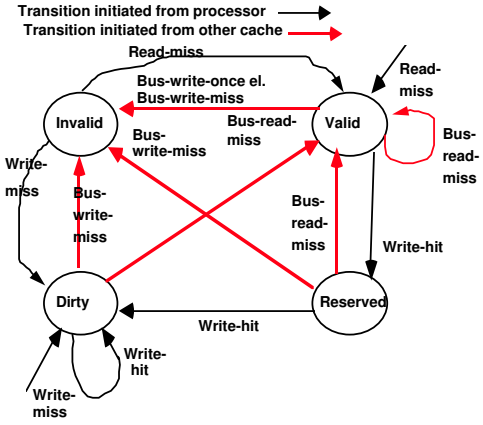
- **Fig. 7.15, del a)**
  - "Tversigjennomskriving" (WT)
  - Hurtigbuffer-blokk , tilstand
    - » gyldig (Eng. valid)
    - » ugyldig (Eng. invalid)
  - Hver blokk bruker samme tilstandsdiagram
  - Notasjon:
    - » R(i), W(i), Z(i) betyr at "egen prosessor" leser, skriver eller erstatter blokken
    - » R(j), W(j), Z(j) betyr at "en annen prosessor" leser, skriver eller erstatter *sin* kopi av den samme (hovedlager-)blokken
- **Fig. 7.15, del b)**
  - Utsatt tilbakeskriving (WB)
  - Hurtibuffer-blokk, tilstand
    - » RO (read-only)
    - » RW (read-write)
    - » INV (invalid)
  - Eierskapsprotokoll, en blokk kan eies av
    - » hovedlager; mange buffere kan være i tilstand RO for denne blokken
    - » en blokk, tilstand RW. Tilstand INV i alle andre hurtigbuffere

Goodman´s "Write-once"-protokoll

- **Den kronologisk første (1983)**
  - Skriv-ugyldiggjør buss-snusing protokoll
- **Kombinasjon av "write-back" og "write through"**
  - WT første gang en skriver til en blokk
    - ⇨ navnet "Write-once"
    - ⇨samtidig ugyldiggjøres evt. andre kopier
  - WB ved ytterliggere skriving til blokk
  - trenger en egen tilstand, RESERVED
- **Tilstander;**
  - *INVALID: inconsistent (se under)*
  - VALID: clean, potentially shared, consistent with memory
  - RESERVED: written once, guaranteed only copy in any cache, consistent with memory
  - DIRTY: written more than once, only copy in the memory system

OBS: [Hwa93] skriver at 'invalid' også representerer ikke-i-hurtigbuffer. Dette stemmer ikke med artikkelen til Stenstrøm som Fig. 7.16 er hentet fra. Blokker som ikke er lagret i hurtigbufferet har en udefinert tilstand. Dette tilsv. transisjoner "utenifra" i tilstands-diagrammet.

Goodman´s "Write-once"-protokoll Tilstandsdiagram



INVALID: inconsistent
VALID: clean, potentially shared, consistent with memory
RESERVED: written once, guaranteed only copy in any cache, consistent with memory
DIRTY: written more than once, only copy in the memory system

Goodman's write-once beskrivelse

read-hit fører aldri til endret tilstand og tegnes ikke inn

read-miss (blokk ikke inne el. 'invalid')
***1** ingen cache har kopi i tilstand 'dirty': Hovedlager har da konsistent kopi. Prosessor henter fra hovedlager, den nye blokken får tilstand 'valid'. Alle kopier av blokken, er konsistente, og endrer tilstand til 'valid' ved en transisjon som kalles 'Bus-read-miss'. (Kalles Read-Blk i fig 7.16).
***2** en (annen) cache har en kopi i tilstand 'dirty': Denne blokken er blitt skrevet til og er eneste kopi. Eier-prosessor av denne forhindrer hovedlager i å sende blokken, over-sender sin egen kopi, og oppdaterer hovedlager. Blokk i cache som oversender vil endre tilstand til 'valid'.

write-hit
***3** Blokk er 'dirty': den er eneste riktige kopi, og skriving kan fortsette.
***4** Blokk er 'reserved': men endrer da tilstand til 'dirty' fordi vi har skrevet mer en enn gang til blokken.
***5** hvis blokk er 'valid', skriver en til hovedlageret (= write-once) og endrer tilstand til 'reserved'. Alle andre kopier av denne blokken ugyldiggjøres ved en ugyldiggjørings- kommando 'Buss-write-once' (invalidate) som kringkastes.

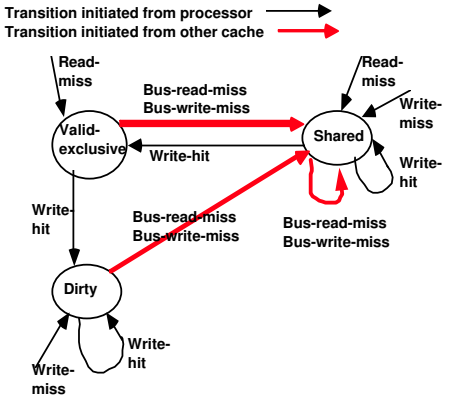
write-miss
***6** (Som read-miss) Ingen cache har kopi i tilstand 'dirty'. Blokken lastes fra hovedlager som er konsistent, prosessor skriver i blokken som blir eneste riktige kopi, dvs. tilstand 'dirty'. Andre prosessorer som evt. har kopi oppfatter dette som en transisjon 'Bus-write-miss' (kalt 'Read-Inv' i fig. 7.16), og må ugyldiggjøre sin kopi.
***7** En (annen) cache har kopi av blokken i tilstand 'dirty': Eier-prosessor oversender sin (eneste riktige) kopi, og endrer egen tilstand for blokken til 'invalid'. Mottaker-prosessor setter blokken til 'dirty', fordi den er eneste riktige kopi.

Blokk-erstatning Det er bare blokker i tilstand 'dirty' som må skrives ut før de erstattes, ellers er hovedlager konsistent.

DEC´s "Firefly"-protokoll

- **Firefly, multiprosessor arbeidsstasjon fra DEC**
- **Skriv-oppdater buss-snusing protokoll (Eng. write-update)**
  - Alle hurtigbuffer-kopier er konsistente, har derfor ikke tilstanden 'invalid'
- **Kombinert WT/WB, dvs.**
  - utsatt tilbakeskriving (WB) for private blokker
  - tvers-igjennomskriving (WT) for delte blokker
- **En dedikert busslinje, ´SharedLine´, benyttes for å oppdage deling**
- **Tilstander;**
  - VALID-EXCLUSIVE (only cached copy, consistent with memory)
  - SHARED (consistent with memory)
  - DIRTY (only cached copy, dirty, inconsistent with memory)

DEC´s "Firefly"-protokoll Tilstandsdiagram



*** VALID-EXCLUSIVE (only cached copy, consistent with memory)**

*** SHARED (consistent with memory)**

*** DIRTY (only cached copy, dirty, inconsistent with memory)**

DEC's Firefly beskrivelse

Read-miss

***1** Ingen hurtigbuf. har kopi: blokk hentes fra lager, -> tilstand 'valid-ex'.

***2** En eller flere andre hurtigbuffere har blokken: **a)** Hvis en slik blokk er ´dirty´, er den alene. Den oppdaterer da hovedlager og oversender blokken. **b)** Hvis en slik blokk er ´valid-excl.´, er den alene, oversender blokken, men trenger ikke oppdatere hovedlageret. **c)** Hvis flere andre hurtigbuffere har blokken, er de alle i tilstand ´shared´, og de oversender blokken. Den nye blokken, og den aktuelle blokken i alle de hurtigbuffere som hadde kopi endrer tilstand til ´shared´. (Merkes med Read-miss, og Bus-read-miss)

Write-hit

***3** Blokk er ´dirty´, kan fortsette uten endring.

***4** Hvis blokken er ´valid-excl.´, kan en utføre skriving umiddelbart men må skifte tilstand til ´dirty´.

***5** Blokk er ´shared´, hovedlager og alle andre ´shared´-kopier oppdateres samtidig. Hvis deling av blokk fortsetter, indikeres dette av ´SharedLine´, og en forblir i tilstand ´Shared´.

***6** Hvis deling av blokk har opphørt, indikert av ´SharedLine´, trenger vi ikke kringkaste oppdateringer lenger, og endrer tilstand til ´valid-excl.´

(detaljer for bruk av 'Shared Line' gis senere)

Write-miss

***7** Hvis blokken ikke finnes i noe annet hurtigbuffer, hentes den fra hovedlager og endres. Blokk får tilstand ´dirty´.

***8** Hvis blokk finnes i et annet hurtig-buffer, blir denne oversendt og den nye blokken satt i tilstand ´shared´. Mottaker-prosessoren skriver til den nye blokken, og oppdaterer hovedlager. Andre hurtigbuffere med den samme blokken i tilstand 'shared' vil se dette og gjøre den samme oppdateringen. Blokk som sendte over etterspurt blokk går over i tilstand ´shared´uansett hvilken tilstand den var i. (Merkes Write-miss og Bus-write-miss)

Kilder

- **Kilde for detaljerte studier;**
  - ***Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model,*** av James Archibald og Jean-Loup Baer, ACM Transactions on Computer Systems, nov 86, side 273-298.
    - » DEC Firefly, bruk av 'SharedLine':
    - » Skrive-treff for prosessor som har blokken i tilstand 'shared': Skriving venter til en "får" bussen og kan oppdatere hovedlageret. Andre hurtigbuffere som evt. framdeles har kopi av denne blokken observerer dette og oppdaterer sin egen kopi (med det ene ordet som skrives). I tillegg aktiviserer de 'SharedLine'. Prosessor som skriver i eget hurtigbuffer kan da avgjøre om deling har opphørt ved å se på denne linjen. Dette avgjør hvilken tilstand en går til.
  - ***A Survey of Cache Coherence Schemes for Multiprocessors***, av Per Stenström, IEEE Computer, juni 1990, side 12-24.
  - ***Synchronization, Coherence and Event Ordering in Multiprocessors***, av Michel Dubois, Christoph Scheurich og Fayé A. Briggs, IEEE Computer, februar 1988, side 9-21

Bruk av kilder

en nyttig digresjon

- **1 divisjon, f.eks. ACM og IEEE Transactions on .....**
- **2 divisjon, f.eks. IEEE Computer, og Comm. of ACM**
- **3 divisjon, f.eks. BYTE,**
- **verdien av å lese flere kilder om samme tema**
  - ulike vinklinger
  - ulike forståelser
  - ulike framstillinger
  - ulike feil
- **verdien av å "nøste seg bakover" mot primærkilden**
  - (fra lærebok til original-arbeid)
  - færre kilder til feil
  - men tiden går.....
  - "slurv & feil" --> vær alltid på vakt

Koherens-protokoll og ytelse

- **Motivasjon for buss-snusing**
  - redusere lager-trafikk
  - redusere gj.snittlig aksess-tid
- **Faktorer som påvirker ytelse**
  - programlast (Eng. "workload")
  - protokoll
  - implementasjon av protokoll
  - blokk-størrelse
- **Buss-trafikk øker med grad av deling av data**
- **Dårlig skalerbarhet**

Koherens protokoll, realisering

- **Tilstandsmaskinen for koherens-protokoll realiseres i hurtigbuffer-kontroller**
- **Hurtigbuffer-katalog lagrer tilstand til hver blokk**
  - aksesseres fra prosessor og buss-kontroller side
    - » (Alternativt: for katalog-metoden kan tilstands-info. også lagres sammen med hovedlageret)
  - kan få kollisjoner, duplisering kan benyttes
- **Noen protokoller krever egne buss-linjer**
  - spesielle buss-standarder
  - f.eks. IEEE Futurebus, SCI

Katalog-metoden

Kap. 7.2.3

- **Sammenkopling av et større antall prosessorer til delt lager gjør bussen til en flaskehals**
  - **Skriv-ugyldiggjøring** øker lagertrafikk fordi det medfører økende andel lese-bom (nødvendig)
  - Skriv-oppdater øker bussbelastning fordi det medfører unødvendige oppdateringer
- **=> Flertrinns sammenkoplingsnettverk**
  - Kringkasting er meget kostbart i fletrinnsnettverk
  - Ønskelig å benytte selektiv kringkasting (Eng. multicast)
    - » dvs. til bare de hurtigbuffere som trenger beskjed
- **=> Katalog-metoden**
  - Parametre:
    - » hvilken informasjon lagres
    - » hvor lagres informasjonen
    - » hvordan vedlikeholdes informasjonen
  - Sentral katalog
    - » Tang, 1976
    - » stor, assosiativt søk
    - » flaskehals, kostbar
  - Distribuert katalog
    - » Censier og Feautrier, 1978
    - » Fig. 7.18

Komplette og begrensete hurtigbuffer-kataloger

- **Komplette kataloger**
  - holder oversikt over alle kopier av en blokk
  - krever $O(mn)$ lagerplass
    - » m = # ord i lager, ofte $O(n)$
    - » n = # prosessorer
    - » Fig. 7.19 a)
- **Begrensete kataloger**
  - oversikt over kopi i et begrenset antall hurtigbuffere
  - lagrer prosessor-pekere
  - Over grensen?
    - » Alternativ 1:
      - tillates ikke
      - $Dir_i NB$
      - peker-overtakelse (Eng. eviction)
      - fig. 7.19 b), viser $Dir_2 NB$
    - » Alternativ 2:
      - benytter kringkasting, $Dir_i B$
  - $O(m \log n)$ lagerplass

Lenket hurtigbuffer-katalog

- **Plassbesparende realisering av komplett (el. begrenset) katalog**
  - bruker ikke mere plass enn nødvendig
  - MEN, selektiv kringkasting kan bli langsom
- **Enkel lenket liste**
  - eks. Fig. 7.19 c)
  - erstattning av blokk ("fortrenges")
    - » Alt.-1: lenke blokk ut av kjeden
    - » Alt.-2: ugyldiggjøre alle etterfølgende blokker
- **Dobbelt lenket liste**
  - økt overhead og kompleksitet
  - raskere blokk-erstatning
  - eks. SCI, Scalable Coherent Interface
    - » Dolphin
    - » UiO, programverifikasjon, formelle bevis
    - » Kap. 9.1.4

Koherens løsninger, fordeler og ulemper (oppsummering)

1) Delte hurtigbuffre
- (+) usynlig for bruker og OS
- (-) reduserer ikke andel av lager-konflikter
- (-) gir langsommere hurtigbuffer

2) Buffre "ufarlige" data
- (-) synlig for bruker el. kompilator
- (-) fragmentering
- (-) ineffektiv håndtering av I/O

3) Buffre "ufarlige" og beskyttede data
- (-) "cache-flush", må enten tømme hele hurtigbufferet, el. delte data må merkes

4) Katalog-metoden
- (-) global tabell kan bli flaskehals, forsinker aksesser

5) Buss-snusing
- (-) snusing og protokoll gjør busskontroller langsommere
- (-) bruk av buss begrenser #prosessorer

Synkronisering Grunnleggende begreper

- **primære maskinvare funksjoner**
  - nødv. for å kunne garantere korrekt kommunikasjon og synkronisering mellom prosesser/prosessorer
- **sekundære maskinvare funksjoner**
  - forenkler konstruksjon av operativsystem og parallelle applikasjoner
- **kommunikasjon**
  - utveksling av data mellom prosesser
- **synkronisering**
  - spesiell form for kommunikasjon der data er kontroll-informasjon
  - Typisk bruk:
    - » a) sørge for riktig rekkefølge av hendelser i samarbeidende prosesser
    - » b)sørge for eksklusiv adgang til delte ressurser

Synkroniseringsmekanismer

• På maskinvare-nivå
- atomisk lager-aksess (og vanskelige algoritmer)
- Test& Set, Compare& Swap o.l.
- Fetch& Add
- Busy-Waiting, spin-lock (aktiv venting)
- Suspend-lock, bruker avbrudd for å vekke opp prosesser som sover i "vente-kø" (passiv venting)

("Wired barrier" er ikke pensum)

• I programvare
- semaforer (P og V)
- barriere-synkronisering
  - = "join" av et antall parallelle prosessorer
  - via meldingsutveksling
  - via spesial-maskinvare

+ mangle flere primitiver

Multimaskiner

- **Multiprosessorer**
  - uten delt lager
  - meldingsutveksling
- **Sentrale konstruksjonsvalg, Fig. 7.22**
- **Tre generasjoner av multimaskiner**
  - Tabell 7.1
  - 1ste generasjon (1983-87)
    - » Cosmic Cube, Caltech, Charles Seitz, 1983
    - » "store-and-forward" ruting
  - 2ndre generasjon (1988-92)
    - » Intel (Touchstone) Delta (prototype for Intel Paragon)
    - » "wormhole routing", HW-støttet ruting
    - » Fig. 7.23
  - 3dje generasjon (1993-97)
    - » Mosaic C. (Caltech), J-Machine (MIT), (Kap. 9)
    - » fin-kornet multimaskin
    - » globalt delt virtuelt lager
- **Trender**
  - flere prosessorer
  - kraftigere noder
  - større kommunikasjonsbåndbredde
  - lavere kommunikasjonsforsinkelse

PTUF (en digresjon)

PTUF = Parallellteknologi i undervisning og forskning ved NTH og SINTEF

Koordinering av NTH/SINTEF
- samarbeid internt
- stå samlet utad mot bl.a.
  - . Forskningsrådet
  - . EF forskningsprosjekter
- IKU, og STATOIL også med

Bl.a. anskaffelse av parallell datamaskin for kompetanseoppbygging

Ønsker flere brukere av maskinen

I fag. 45214 (miniprosjekt)
- studie av arkitektur
- målinger av sentrale maskin-karakteristika
- vanlig bruk av maskinen er normalt ikke nok for å bli godkjent som "krevende nok"

Intel og superdatamaskiner

• Intel
- 25 år
- SSD = supercomputer systems division (noe utviklingsarbeide i Europa)
- levert superdatamaskiner siden 1987
- relativt konservativt firma

• Intel superdatamaskiner

De første hyperkuber - nå 2D rutenett

- iPSC/1 (CalTech Cosmic Cube)
- iPSC/2
- iPSC/860 (gamma)
- Touchstone Delta, 12 Gflops (32 peak)
- intel PARAGON (sigma)

TeraFlop maskin forventes på en rekke applikasjoner i 1995

Intel Paragon

* MIMD
* medium # prosessorer (< 1000)
  - Fig. 7.24
* mesh = 2D rutenett
  - hurtig datakomm. i HW

* Trondheim des 92:
  - PARAGON XP/S - A4
  - 56 prosessorer (et kabinett)
  - ansvarlig (PTUF), Tor Helge Hansen (Sima)

* Front panel visualisering
  - viser "hovedtrekk" i data.komm. mellom noder grafisk
  - kamera og monitor på kontorer !

MIMD programmering på PARAGON

• Programmerer ser "et komplett sammenkoplet nett av noder", dvs. systemet ordner rutingen

• Velprøvde bibliotek for ulike typer meldingsutveksling
- egne
- PVM, PICL
- PARMACS (Argonne macros)
- ... standarder ---> portabilitet

• Viktig at algoritmer er parametriserte m.h.p. # prosessorer som benyttes
`(mynode, numnodes)`

• Applikasjon *kan* deles opp i "host"-del og node del, der "host"-del kan kjøres på en service-node eller ekstern arbeids-stasjon

• ***"programming is different, not difficult"***

Paragon - arkitektur

2D rutenett av noder koplet sammen via bakplan (17 lags kort)
- Fig. 7.25

2 x i860-XP prosessor og iMRC (Mesh routing chip) i hver node. Applikasjons-prosessor og meldingsprosessor

iMRC håndterer meldingstransport,
- videresender 1 ord på 40 ns, 80 ns rundt hjørner
- kan i teorien samtidig "ekspedere", 1GB/s (alle kanaler)
- 325 kontakters kundespes. VLSI, CMOS 0.65 micron
- Fig. 7.26

(Rykter: likevel blir dette for langsomt for store maskiner. Neste maskin vil muligens ha ett flatt hierarkisk nettverk?)

maskinens noder er delt inn i
- beregningsnoder
- servicenoder
- I/O noder

Paragon - arkitektur, forts.

- **programkode for meldingshåndtering ligger i hurtigbufferet til meldings-prosessor**
- **400 MB/s mot lokalt kager**
- **I/O**
  - skalerbart og parallelt masse-lager
  - RAID
- **Neste maskin vil bruke 64 bits etterfølger til i860 XP, 4 applikasjons-prosessorer og en meldingsprosessor på hver node**

(Lasse har mere materiale for de som er interesert)

Paragon, programvare

- **NX/2 OS**
  - UNIX kjerne for node-programmer
- **Bruker ser komplett UNIX ved login**
- **CFS = Concurrent File System**
  - I/O behandles av dedikerte noder
- **Flere applikasjoner kjører samtidig (tidsdelt) på ulike og/eller overlappende partisjoner**
- **"Gang scheduling":** en typisk parallell applikasjon vil dra fordel av å kjøre alle prosessene samtidig, og lenge.
- **MACH microkernel (CMU), OSF/1**
- **programutvikling on-line eller off-line**
- **maskin-deling ("Space-sharing")**
  - velg hvilke deler av maskinen som du vil kjøre applikasjonen på

FORTRAN / "Shared Virtual Memory" SVM

- **HPF = High Performance Fortran**
  - "tabell-fokusert" (Eng. array-based)
  - data-parallelt språk
  - fra arbeids-stasjoner til super-datamaskiner (= skalerbar SW)
  - basert på FORTRAN 77, FORTRAN 90, FORTRAN D, Vienna FORTRAN
  - HPFF
- **FORTRAN 90**
- **STRAND 88, LINDA**

- **Shared Virtual Memory (SVM)**
- **Virtuelt adresserom deles av flere prosessorer, mens lageret er fysisk distribuert**
- **tilbys via External Memory Manager (XMM) i Mach microkernel**
- **økt overhead, men lettere programmering**

Intel og MPP (1992)

- **MPP = Moderatly Parallel Processing?**
- **1 TeraFlop i applikasjoner i 1995**
- **Maskiner av typen PARAGON**
- **Standard språk er viktig, HPF**
- **Intel har mange brukere som ikke "vil gi slipp på meldingsutveksling"**

Meldingsutveksling

- **Kap. 7.4, side 374-393**
  - godt beskrevet i boka
  - selvstudium
  - + evt. rask gjennomgang
- **Meldingsformat**
  - melding, pakke, "flit" ("flow control digit"), (Fig. 7.27)
- **Rutings-strategi, Fig. 7.28**
  - "store-and-forward" (SF)
  - "wormhole routing (WH)
    - » asynkron pipelining, Fig. 7.29
  - tidsforsinkelse (Eng. latency), Fig. 7.30
    - » typisk T_{SF} = 2-6 milli-s, T_{WH} = 5 mikro-s.
- **Virtuelle kanaler (Fig. 7.31)**
- **Vranglås, Fig. 7.32**
  - buffere
  - kanaler
  - sirkulær venting, fig. 7.33
    - » kan brytes v.h.a. virtuelle kanaler

Strategier for flytkontroll

- **Hva gjør vi ved pakke-kollisjoner?**
  - Fig. 7.34
    - » buffring i node, (venter)
    - » blokkering
    - » forkasting og retransmisjon
    - » omkjøring (Eng. detour)
    - » kombinasjoner av disse
- **Deterministisk ruting**
  - kommunikasjonsvei er gitt av sender og mottaker adr.
  - E-kube ruting i hyperkube, Fig. 7.35
  - X-Y ruting i 2D rutenett, Fig. 7.36
- **Adaptiv ruting**
  - velger vei ut i fra tilstand i nettverket
    - » unngå venting el. vranglås
    - » fordele trafikk

Mer om meldingsutveksling

- **Kommunikasjons-mønstre**
  - en-til-en ("unicast")
  - en-til-alle, kringkasting
  - en-til-mange, selektiv kringkasting (Eng. multicast)
  - mange-til-mange, "konferanse"
- **Ytelsesparametre for ruting**
  - forsinkelse (Eng. latency)
  - trafikk
  - avveining mellom disse, Fig. 7.38
- **Hyperkube**
  - kringkasting, Fig. 7.39a)
  - selektiv kringkasting, Fig. 7.39b)
- **Virtuelle nettverk**
  - Fig. 7.40
- **Partisjonering av nettverk**
  - Fig. 7.41