Kap. 2: Sentrale egenskaper i program og nettverk

- **Betingelser for parallellitet**
  - data, kontroll og ressursavhengigheter
  - Bernstein's betingelser
  - maskinvare og programvare parallellitet
  - kompilatorens rolle
- **Programoppdeling og tidsplanlegging (Eng. scheduling)**
  - nivåer av parallellitet
  - "grain packing" og tidsplanlegging
  - tidsplanlegging av oppgaver på multiprosessor
- **Mekanismer for programflyt**
  - kontroll-flyt
  - data-flyt
  - etterspørselsdrevet (Eng. "demand driven")
- **Arkitekturer for sammenkopling av/i systemer (Eng. systems interconnect architecture)**
  - grunnleggende begreper, grafteori
  - statiske nettverk
  - dynamiske nettverk

Data-, kontroll- og ressurs-avhengighet i beregninger

- **Parallell utførelse krever uavhengighet**
- **Hva er en beregningsenhet?**
- **Avhengighetsgraf**
  - analyse av denne viser muligheter for parallellitet
- **Slike avhengigheter gir (heldigvis) bare en partiell ordning**
- **Kompilatorer (og systemkonstruktører) analyserer avhengighetsgrafen (beregningsgrafen) og forsøker å utnytte uavhengigheter**
  - omstrukturerering av et program (en beregning) kan gi større grad av uavhengighet
    - » redundans
    - » lokale operasjoner er fordelaktig
  - spesiell programmerings-stil og kompilatordirektiver kan gjøre analysen lettere for kompilatoren

Data-avhengighet

S1 og S2 er (f.eks) setninger i et program

❶ **flyt-avhengighet** S1 → S2
- et resultat av S1 kan påvirke S2

❷ **anti-avhengighet** S1 ⊣→ S2
- S2 må følge etter S1 i programmet fordi et resultat av S2 påvirker S1 (ikke progr. hensikt)

❸ **utdata-avhengighet** S1 ○→ S2
- S1 og S2 skriver til samme utdata-variabel

❹ **I/U-avhengighet** S1 —I/U→ S2
- f.eks. avhengighet knyttet til bruk av samme fil (ikke variabel)

❺ **ukjent avhengighet** S1 —?→ S2
- tilfeller der det er for vanskelig el. umulig å avklare evt. avhengighet --> må anta avhengighet

Kontroll- og ressurs-avhengighet

- **Kontroll-avhengighet**
  - tilfeller hvor rekkefølge av setninger (og dermed avhengigheter) ikke kan avgjøres før ved kjøretid
- **Ressurs-avhengighet**
  - konflikter ved bruk av delte ressurser
    - » funksjonell enhet (i ALU), ALU/CPU, registre, lager-plass, (bånd-stasjon) etc.

Bernstein's betingelser

- **Hvilke prosesser kan utføres samtidig?**
- **Prosess P_i**
  - inndata mengde I_i
  - utdata mengde O_i
- **P_1 og P_2 kan utføres i parallell, $P_1 \parallel P_2$**

$$I_1 \cap O_2 = \varnothing$$
$$I_2 \cap O_1 = \varnothing$$
$$O_1 \cap O_2 = \varnothing$$

  - dvs. P_1 og P_2 kan utføres samtidig hvis de er flyt-uavhengige, anti-uavhengige og utdata-uavhengige
  - vi får samme resultat om de utføres samtidig eller serielt i vilkårlig rekkefølge
- **Et antall prosesser P_1, P_2, ... P_k kan utføres samtidig hvis Bernstein's betingelser gjelder for alle par $P_i \parallel P_j$, der $i \neq j$**
- **Eks. 2.2, fig. 2.2**

Maskin- og programvare-parallellitet

- **Utnyttelse av parallellitet krever**
  - teoretiske betingelser tilfredsstilt
  - parallell maskinvare tilgjengelig
  - samspill programvare/maskinvare
    - » f.eks. kunne avbilde logisk parallellitet på fysisk parallellitet
- **Parallellitet i programvare påvirkes av**
  - type av problem/applikasjon
  - instans av problem/applikasjon
  - løsning/algoritme
  - programmerings-språk og -stil
  - kompilator og RTS
- **Eks. 2.3, fig. 2.3**

Parallelliserende kompilatorer

Ond sirkel for parallelliserende kompilatorer

Sirkelen kan brytes ved å konstruere maskinvare (arkitektur) og kompilator (arkitektur) samtidig og i samarbeid

Programoppdeling og tidsplanlegging (Eng. scheduling)

Begreper/terminologi

- **Kornstørrelse** (Eng. grain size)
  - (Prosess-størrelse, grovhet)
  - (Kornethet el. granularitet), (Eng. granularity)
  - et mål på mengden av beregninger i en prosess
    - » f.eks. # instruksjoner
    - » fin, medium, grov
- **Forsinkelse** (Eng. latency)
  - kommunikasjons-overhead mellom del-systemer
  - lager-forsinkelse
  - synkroniserings-forsinkelse
- **Nivåer av parallellitet, fig. 2.5.**
- **Kornstørrelse og forsinkelse er relatert**
  - større "korn" kan gi mindre forsinkelse pr. enhet
  - større "korn" kan redusere mulig parallellitet
  - ønskes balansert

"Grain packing & scheduling"

- **Avveining** (Eng. trade-off) **mellom parallellitet og kommunikasjons-overhead (inkl. synkronisering)**
- **Hvilken kornstørrelse er optimal for et gitt program på en gitt maskin?**
  - hva er optimalt?
    - » minimum tid
      - et program raskest mulig
    - » maksimal ytelse
      - flest program pr. tidsenhet
    - » minimum kostnad
      - hva er kostnad?
        - (ProsessorBruk) x (Tid)
        - (HW-areal) x (Tid)
        - Kommunikasjons-kostnad + Prosesseringskostnad

Tidsplanlegging på multiprosessorer

- **"Scheduling" (= tidsplanlegging)**
  (= kjøretidsplanlegging)
  (= kjøreplanlegging)
  (= fordeling, i.flg. Norsk Dataordbok)
- **En optimal tidsplan (kjøreplan) kan generelt ikke finnes i polynomisk tid**
  - unntak, enkle spesialtilfeller
  - heuristiske algoritmer kan finne nesten-optimale løsninger i polynomisk tid
- **Duplisering av beregninger (dvs. økt redundans) kan gi redusert ventetid og kommunikasjons-forsinkelse, eks. fig 2.8**
- **Kombinert "grain packing" og kjøreplanlegging (eks. 2.5)**
  1) konstruer en finkornet beregningsgraf, FG
  2) lag kjøreplan for FG
  3) utfør "grain packing" på FG, det gir GG
  4) lag kjøreplan for GG

Mekanismer for programflyt

- **Konvensjonelle maskiner (kontroll-flyt)**
  - eksplisitt kontroll av programflyt gjennom progr.-teller
    - » iboende sekvensiell
    - » side-effekter
  - parallell kontroll-flyt
    - » spesial-språk og/eller flere instruksjonsstrømmer
    - » mest utbredt i dag
- **Dataflyt-maskiner (data-flyt)**
  - programflyt avgjøres implisitt ved tilgjengelighet av data (dvs. data-drevet beregning)
  - "eager evaluation"
  - ikke side-effekter
  - iboende parallell, fin-kornet
  - stort potensiale for massivt parallelle beregninger
  - mangel på effektive maskiner
  - mye forskning gjort over lengre tid
- **Reduksjons-maskiner (etterspørsels-drevet) (Eng. "demand driven")**
  - en beregning utføres når dens resultat etterspørres
  - "lazy evaluation"
    - » egnet for funksjonelle språk
  - to hovedtyper
    - » streng reduksjon
    - » graf-reduksjon

Arkitekturer for sammenkopling av systemer

- **Statiske nettverk**
  - direkte, faste oppkoplinger
  - eks. ring, hyperkube, m.m.fl.
- **Dynamiske nettverk**
  - linjer som går via styrte svitsje-elementer
  - eks. (1) buss, (2) fler-trinnsnettverk, (3) "krysskoplingsnett" (Eng. "cross-bar")
- **Viktige egenskaper**
  - funksjonalitet
  - forsinkelse
  - båndbredde
  - maskinvare kompleksitet
  - skalerbarhet

Sammenkoplingsnettverk, sentrale begreper

- **Nettverks-størrelse**
  - antall noder
- **Node-grad, inn-grad, ut-grad**
  - påvirker en nodes kostnad
- **Diameter**
  - den lengste korteste vei mellom to noder, dvs.
    $l_{i,j}$ = lengden av korteste vei mellom node i og j
    $S =$ diameter
    $S = \max(\forall i, j \ : \ l_{i,j})$
  - Nye teknikker for meldingutveksling basert bl.a. på pipelining gjør at stor diameter har mindre betydning
- **Tverrsnitt-bredde**
  - i # kanaler
  - i # bit-linjer
- **Symmetrisk topologi**
  - lettere å bruke
  - mer skalerbar
- **(Data-) rutingsfunksjoner**
  - f.eks. skift, roter, kringkast

Mer om nettverk

- **Nettverks-ytelse**
  - "antall meldinger nettverket kan håndtere pr. tidsenhet"
    - » meldingslengde ?
    - » "håndtere"?
- **Nettverks-kapasitet**
  - antall meldinger som kan "være i nettverket" på en gang
- **"Flaskehals-par" (Eng. hot-spot)**
  - et par av noder som står for en uforholdsmessig stor del av lager-trafikken
- **Flaskehals-ytelse**
  - maksimal rate for sending av meldinger mellom to gitte noder
- **Universelle nettverk**
  - kan effektivt simulere andre nettverk
    - » "embedding"
    - » simulering
  - f.eks. "fat-tree" og "k-nære n-kuber"
- **----------**
- **Tabell 2.2.: oppsummering**
  - og mange flere finnes

Dynamiske nettverk

- **Velegnet for generelle maskiner, eller applikasjoner med varierende kommunikasjonsbehov**
- **Hovedløsninger**
  - etter økende kostnad og ytelse:
  1) buss
  2) fler-trinns-nettverk
  3) krysskoplings-nettverk (Eng. cross-bar)