Stanford FLASH Multiprosessor

- **Integration of Message Passing and Shared Memory in the Stanford FLASH Multiprosessor**
  - FLASH = FLexible Architecture for SHared Memory
  - Multiprosessor basert på meldingsutveksling
  - John Heinlein et.al, ASPLOS VI, 1994.
- **Mål**
  - **Fleksible mekanismer for kommunikasjon**
    - » **Kombinere *delt-minne* med *meldingsutveksling* innen en arkitektur**
      - begge typer er best i noen tilfeller
      - hvordan unngå (for mye) økt kompleksitet i en slik hybrid løsning?
    - » Tilby **mange** forskjellige **protokoller**
      - ulik meldingsstørrelse
        - fra synkronisering til video-overføring
      - ulik kompleksitet
        - fra "memory copy" til store standarder som MPI (Message Passing Interface)
      - ulik funksjonalitet "active messages"
        - fra "Fetch&Op" til RPC
- **300 MB/s data overføring var forventet**
  - Hva oppnådde de?

Effektiv meldingsutveksling

- **Hvordan redusere meldings-forsinkelse (Eng. latency) og øke båndbredde**
  - **unngå kostbare system-kall** ved å tilby bruker direkte aksess til det delsystemet som håndterer meldinger
    - » spesielle lese/skrive operasjoner mot kontroller
    - » nødv. beskyttelse mot andre brukere/prosesser må ikke ødelegges
  - **unngå kopiering av data**
    - » sende pekere; men dette krever at bruker-prosess og meldingssystemet kan aksessere det samme adresse-området
  - **overlapp beregninger** (Eng. computation) med kommunikasjon
    - » avlaste prosessoren
    - » færre avbrudd til prosessor

Flash - Arkitektur

- **Node**
  - 'Off-the-shelf' Mikroprosessor + hurtigbuffer
  - Minne (del av distribuert delt minne)
  - **MAGIC** (Memory and General Interconnect Controller)
  - I/U delsystem
  - Nettverk komponenter
- **2D rutenett (Eng. mesh)**
- **Figur (fra annen artikkel)**

Programmerbar node kontroller (MAGIC)

- **Programmerbar node kontroller**
  - Se fig. 1 (a)
  - Kobler sammen: prosessor / minne (del av delt distribuert minne) / nettverk komponenter
  - Prosesserer alle transaksjoner mellom disse komponentene
    - » **ikke** direkte kobling mellom prosessor og minne
  - MAGIC implementerer
    - » meldingsutvekslings-protokoller
    - » hurtigbuffer koherens-protokoller
- **Kontroller kan programmeres om til ulike protokoller**
  - Fordeler
    - » Implementasjon av protokoll-funksjonalitet
      - redusere prosessor overhead
  - Ulempe
    - » Økt kostnad, dvs. billigere å implementere en 'Hardwired' Kontroller for en spesifikk protokoll
    - » Redusert ytelse
      - øket forinkelse for aksess til vanlig (lokalt) minne fra prosessor
      - sentralisert kontroll kan bli flaskehals

MAGIC Arkitektur

- **Se fig. 1 (b)**
- **Parallel behandling av kontroll og data**
  - 'Message Join' komponent
  - 'Message Split' komponent
- **Behandler protokoll-meldinger i pipeline**
  - Inbox
    - » velger adresse for "handler"
    - » evt. spekulativ lager-operasjon
  - PP - Protokoll Prosessor
    - » utfører "handler", kan ikke avbrytes
    - » "dual-issue"
    - » spesial-instruksjoner for protokoll-håndtering
    - » ikke irrelevante instr. som FLOP og avbrudd
    - » Har egen I- og D-cache
      - hurtig aksess til protokoll kode og tilstand
  - Outbox
    - » styrer melding til prosessor el. nettverk
    - » oppdaterer protokoll-tilstand
- **MAGIC inneholder en egen datavei for aksess fra prosessor til lokalt minne**

Kommunikasjon mellom prosessor og MAGIC (1/2)

- **Vanlig 'Memory-Mapped' grensesnitt**
  - Direkte aksess fra bruker-prosesser
    - » starter protokoll-operasjoner
    - » "monitorerer" (overvåker) status
  - Utføres som ikke-cachede skrive og lese-operasjoner til bestemte fysiske adresser
- **4-delt addresse-rom**
  - *Normalt område*
    - » vanlig bruker-aksess til data
    - » håndteres via hurtigbuffer-koherens protokoll av MAGIC
  - *Meldings-område*
    - » aksess hit "trigger" ulike melding-"handlere"
      - **Skriv**: aktuell fysisk adresse gir "kall-type" mens dataverdi gir parameter
      - **Les**: spesifiserer type av respons gjennom aktuell adresse-verdi og forventer svar fra MAGICgjennom returnert verdi
  - *I/U område*
  - *Reservert område*
    - » ikke brukt

Kommunikasjon mellom prosessor og MAGIC (2/2)

- **Kommando-sekvens protokoll**
  - Et antall skrive-operasjoner etterfulgt av en lese-operasjon
  - MAGIC returnerer status
    - » ack
      - OK, kommando godkjent
    - » nack - retry
      - ikke godkjent (f.eks. ikke gyldig, ikke nok ledige ressurser)
    - » kommando utført varsles gjennom egne dedikerte lager-adresser som er avtalt i (fastsatt av) protokollen
- **Protokoll-operasjon datastruktur**
  - **Lagrer (i SW)** deler av kommando og status-informasjon knyttet til denne
    - » overlever prosess-bytte i prosessor
      - MAGIC varsles om skifte av PID
      - en HW-kø ville ikke ha gjort det !
    - » SW-løsning gir også mulighet for å skreddersy strukturen til protokollen
  - Lagres i minne til operasjon er ferdig
  - Aksesseres gjennom eget MAGIC hurtigbuffer

Flash: Addressering

- **Bruker gir virtuell adresse**
- **MAGIC trenger fysisk adresse**
- **Har ikke TLB som del av PP-maskinvare**
  - Hvorfor ikke?
    - » hurtigbuffer koherens protokoll trenger det ikke da MAGIC får fysiske adresser fra prosessor
    - » håndtering av TLB og side-feil (Eng. page-fault) ville ha økt kompleksiteten vesentlig.
    - » en liten maskinvare-TLB er muligens lite effektivt da adresse-referanse-mønstre for PP kan være ganske annerledes enn for en vanlig prosessor
- **Bruker "memory mapped" adresse grensesnitt med "double mapping scheme"**
  - fig 2
    - » OS opererer med et skygge-område hvor det bare er "adresse-rom-bit" som oversettes anderledes enn en adresse fra normal-området

Fysiske adresser i MAGIC, hvordan sikre gyldighet (1/2)

- **Hvordan sikre gyldighet av en fysisk adresse under en protokoll-operasjon?**
  - Må forhindre at OS endrer mappingen; dvs. **lås den virtuelle siden**
    - » under hele programmets levetid
      - fjerner fordelen med demand paging
    - » under hele protokoll-operasjonen
      - vanlig i systemer med DMA
    - » MEN, nødvendig interaksjon med OS gir høy overhead
  - **3 alternativer**
    - » Avveining mellom effektivitet og funksjonalitet
    - » Optimal metode avhenger av protokoll
    - » 1) Holde-igjen teknikk
    - » 2) Ugyldiggjørings teknikk
    - » 3) Optimalisert programvare TLB

Fysiske adresser i MAGIC, hvordan sikre gyldighet (2/2)

- **3 alternativer**
  - Avveining mellom effektivitet og funksjonalitet
  - Optimal metode avhenger av protokoll
  - **1) Holde-igjen teknikk**
    - holder igjen forespørsler om endring i oversetting (TLB -entry)
    - alle minne operasjoner er ferdig
    - alle meldings protokoll operasjoner er ferdig
  - **2) Ugyldiggjørings teknikk**
    - » tillater endret oversetting
    - » ugyldiggjør fysisk adresse (side) i MAGIC
    - » ved neste bruk av denne i MAGIC vil programvare-"handler" oppdage ugyldig tilstand og etterspørre ny oversetting fra hoved-prosessor
  - **3) Optimalisert programvare TLB**
    - » egnet for operasjoner som benytter en oversetting (mapping) mange ganger
      - f.eks. sende en lang melding
    - » oversettelse lagres i protokoll-operasjons struktur
    - » kontakter hovedprosessor via avbrudd hvis den ikke klarer en oversettings-forespørsel

Flash: Beskyttelse

- **MAGIC utfører protokoll operasjoner på vegne av en bruker-prosess**
  - trenger derfor multiprogrammering beskyttelse
- **Minne Beskyttelse**
  - aksess rettigheter sendes over til MAGIC sammen med adresse-oversettelse
- **Gyldighet av meldinger / riktig mottaker**
  - alle meldinger merkes med PID
  - MAGIC kjenner PID til kjørende bruker-prosess
- **Begrense effekter av operasjoner til en gruppe av prosesser (en applikasjon)**

Flash: Dataoverføring

- **Dataoverføringer kan spesifiseres fra brukerprogram gjennom det samme "memory-mapped" grensesnitt**
  - "Operation record" (Protokoll operasjons struktur) lagrer overføringsparametre og vedlikeholder status under overføring
  - Data overføres som en serie av komponenter
    - » Størrelse = "cache-line size" (størrelse på hurtigbuffer blokk)
  - "Handler" kjøre utfører overføringen
    - » sender
      - leser komponent fra minne
      - tilføyer hode
      - sender
    - » mottaker
      - handler lagrer komponent i mottakers minne
  - "Programvare-kø" benyttes for
    - » tidsplanlegge dataoverføringer i flere deloperasjoner (komponenter)
      - "Chunking"; en eller flere komponenter hver gang
      - flettes med hurtigbuffer-koherente les og skriv operasjoner
      - "fairness"
  - Ulempe: overhead ved dette (ingenting er gratis!)

Flash vs. Andre Maskiner

- **Bruker-aksess til meldings-protokoller**
  - 'Memory mapped interface'
    - » **CM-5**
  - register-basert
    - » optimalisert for korte meldinger
      - begrenset båndbredde for store meldinger
    - » stor belastning for hoved-prosessor
    - » MDP - J-machine
- **Intel Paragon**
  - Ekstra prosessor for meldings protokoll prosessering
- **Cray T3D**
  - Støtter meldings-utveksling i ett enkelt adresse rom
  - Ikke hurtigbuffer koherens
  - korte meldinger
    - » 32 byte, via avbrudd hos mottaker-prosessor
  - lengre meldinger
    - » blokk-overføring via DMA
    - » må startes opp via OS-kall
  - MEN, vesentlige forbedringer i CRAY T3E