III. Architectura – Organizare logica si fizica

Continut

- Organizare logica
  - Taxonomia lui Flynn
  - SIMD si MIMD
  - Comunicare
    - Spatiu partajat de date
    - Schimburi de mesaje
- Organizare fizica
  - Context istoric
  - Memorie partajata vs. memorie distribuita

De ce? Cum?

- Existe zeci de arhitectuiri paralele diferite, dintre care:
  - Retele de calculatoare (NoW),
  - Clustere de PCuri,
  - Supercalculatoare masive paralele
  - Multiprocesoare simetrice strand cuplate
  - Statii de lucru multiprocesor

  Etc
- Anumite arhitecturi ofera performante mai bune decat celelalte
  - Notiunea de performanta a unui sistem paralel este teoretica si depinde de tipurile de aplicatii pe care le utilizeaza
- **Organizarea logica** se refera la punctul de vedere a programatorului
- **Organizarea fizica** se refera la organizarea hardware

Organizare logica

- Calculatoarele paralele se impart in doua categoo mari:
  1. **Bazate pe scurgerea controlului** (focus exclusiv asupra acestora!!!)
     - Bazate pe aceleasi principii ca si calculatorul secvential sau calculatorul von Neumann,
     - Instructiuni multiple pot fi executate la un moment dat.
  2. **Bazate pe scurgerea datelor**,
     - Referite cateodata ca fiind “non-von Neumann,”
     - Complet diferite: nu exista un pointer la instructiunile active sau o localizare a controlului.
     - Controlul este distribuit: disponibilittaea operanzilor conduce la activarea instructiunilor.
- Componentele critice ale arhitecturii din punct de vedere al perspectivei utilizatorului:
  1. Modalitatile de exprimare a sarcinilor paralele – referit drept **structura de control**
  2. Mechanismele pentru specificarea interactiunii intre sarcini – **modelul de comunicare**
- Sarcinile paralel pot fi exprimate la diferite nivele de **granularitate**:
  1. O extrema: fiecare program dintr-un set de programe este un sarcina in paralel.
  2. Alta extrema: instructiuni individuale sunt vazute ca sarcini paralele.
  3. Intre extreme: varietate de modele pentru soecificarea structurii de control a programelor si suportul arhitectural al acestora.

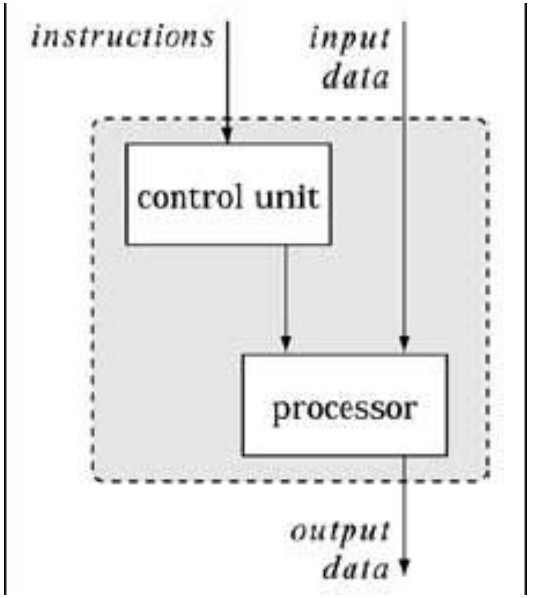
Taxonomia lui Flynn

Staructura de control a platformelor paralele

- M. J. Flynn (1972) a introdus un sistem pentru clasificarea arhitecturilor
  - Valida si astazi si citata in fiecare carte despre calcul paralel
  - Este pe departe cea mai comuna modalitate de caracterizare a sistemelor paralele.
- Clasificare conform nr. de streamuri de instructiuni si streamuri de date
  - un stream este o secventa de instructiuni sau date asupra caruia opereaza un calculator.
- 4 clase de calculatoare bazate pe numarul de
  - Streamuri de instructiuni (singular sau multiplu) si
  - Streamuri de date (singular sau multiplu)
  - ⇒ abbrevieri SISD, SIMD, MISD, and MIMD (pronuntat “sis-dee,” “simdee,” etc)
- Aceasta clasificare este bazata pe relatia intre instructiunile si datele manipulate.
  - ⇒ Este generala sui nu reflecta toate aspectele sistemelor paralele

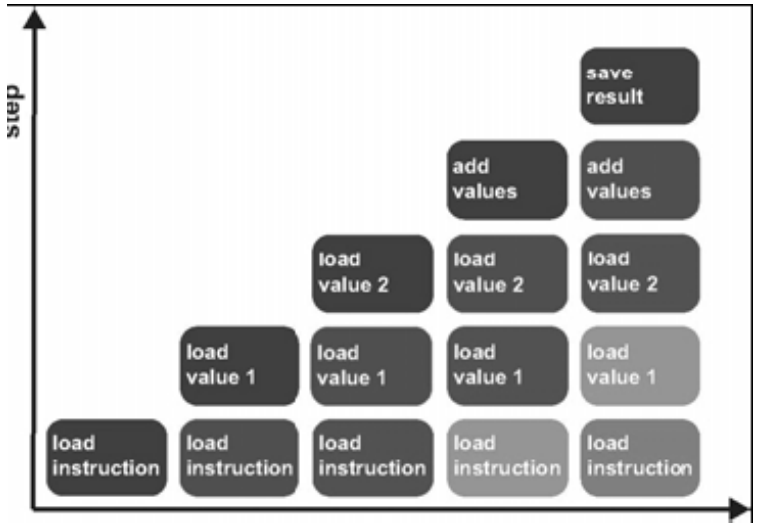
Single Instruction, Single Data (SISD)

- Un flux de instrucțiuni procesează un singur flux de date
- Acesta este modelul obișnuit von Neumann utilizat în practic toate computerele cu un singur procesor.
- Cel mai simplu tip de computer efectuează o instrucțiune pe ciclu
  - cum ar fi citirea din memorie, adăugarea a două valori
  - doar un set de date sau operand
- Un astfel de sistem se numește ***computer scalar*.**
- Examplu: Adunarea a două valori în 5 pași => adunarea a *n* valori în pași 5*n*



SISD si pipeline-uri (conducte)

- În realitate, fiecare dintre pași este de fapt compus din mai multe sub-etape, crescând nr. cicluri necesare pentru o însumare și mai mult.
- Soluția la această utilizare ineficientă a puterii de procesare este ***pipelining:***
  - Dacă există o unitate funcțională disponibilă pentru fiecare dintre cele cinci etape necesare, adăugarea necesită încă cinci cicluri.
  - Avantajul este că toate unitățile funcționale sunt ocupate în același timp, se produce un rezultat la fiecare ciclu.
- Pentru însumarea a *n* perechi de numere, sunt necesare numai (*n* − 1) +5 cicluri.
- Fig. arată însumarea într-o conductă



însumarea a *n* valori

SISD si superscalar

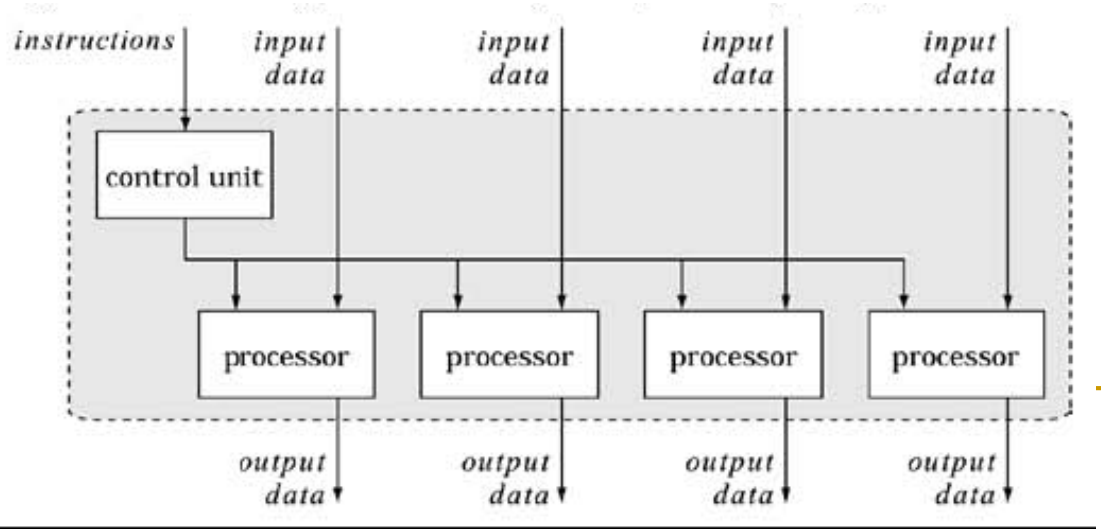
- Deoarece execuția instrucțiunilor durează de obicei mai mult de cinci pași, conductele se fac mai mult timp în procesoare reale.
- Conductele lungi generează o nouă problemă :
  - Dacă există un eveniment de ramificare (*if* -statement),
    - conducta trebuie golită și umplută din nou
    - există un nr. cicluri de ex. la lungimea conductei până la livrarea din nou a rezultatelor.
  - Compilatoarele și procesoarele încearcă, de asemenea, să minimizeze această problemă prin „ghicirea” rezultatului (predicția ramurilor).
- Puterea unui proc. poate fi crescut prin combinarea mai multor conducte
  - Acesta este apoi numit procesor ***superscalar***.
  - Calculele logice (efectuate în ALU - Unitate logică/aritmetică) sunt de obicei separate de matematica in virgula mobile (efectuate de FPU - Floating Point Unit).
  - FPU este de obicei împărțit într-o unitate pentru + și una pentru x.
  - Aceste unități pot fi prezente de mai multe ori, iar unele proc-uri au unități funcționale suplimentare pentru / și calculul rădăcinilor pătrate.
  - Pentru a obține un beneficiu de a avea mai multe conducte, acestea trebuie utilizate în același timp(“pipeline of pipes”)

Multiple Instruction, Single Data (MISD)

- Niciun sistem bine cunoscut nu se potrivește acestei denumiri.
- Such a computer is neither theoretically nor practically possible
- Un astfel de computer nu este nici teoretic, nici practic.
- Unii autori vizualizează MISD ca conducte generalizate în care fiecare etapă efectuează o operație relativ complexă
  - spre deosebire de conductele obișnuite găsite în procesoarele moderne unde fiecare etapă face o operație foarte simplă la nivel de instrucțiuni.
  - un singur flux de date intră în mașină constând din *p* procesoare
  - diverse transformări sunt efectuate pe fiecare dată înainte de a fi transmise la următorul procesor (procesoare).
  - Date succesive pot trece prin transformări diferite
    - declarații condiționale dependente de date în fluxurile de instrucțiuni (controlate)
    - etichete speciale de control sunt purtate împreună cu datele (bazate pe date).
  - Organizarea MISD poate fi privită ca o conductă flexibilă sau la nivel înal cu mai multe căi și etape programabile.
    - Diferența cheie între conducta de mai sus și o arhitectură MISD este aceea că etapele conductelor în virgulă flotantă nu sunt programabile.

Single Instruction, Multiple Data (SIMD)

- Un singur flux de instrucțiuni este difuzat simultan către mai multe procesoare, fiecare cu propriul flux de date.
- Sisteme comerciale: Thinking Machines, MasPar, CPP DAP Gamma II, Quadrics Apemille
- De obicei în aplicații specializate, cum ar fi procesarea digitală a semnalului, care se potrivesc paralelismului cu granulație fină și necesită puțină comunicare între proces.
- Procesoarele vectoriale, care operează pe date vectoriale în mod pipelinat, pot fi, de asemenea, clasificate ca SIMD.
- Exploatarea acestui paralelism este de obicei făcută de compilator.



Un caz SIMD: Vector computer

- Un computer care efectuează o instrucțiune pe mai multe seturi de date se numește ***computer vectorial***.
- Calculatoarele vectoriale funcționează la fel ca computerul scalar pipeline
- Diferența este că în loc să prelucreze valori unice, vectorii de date sunt prelucrați într-un ciclu.
- Numărul valorilor dintr-un vector este limitat de designul CPU-lui.
  - Un procesor vectorial care poate lucra simultan cu 64 de elemente vectoriale poate genera, de asemenea, 64 de rezultate pe ciclu
- Pentru a utiliza efectiv performanța teoretic posibilă a unui computer vectorial, calculele în sine trebuie să fie vectorizate.
- Exemplu:
  - Fie următorul segment de cod care adaugă 2 vectori :

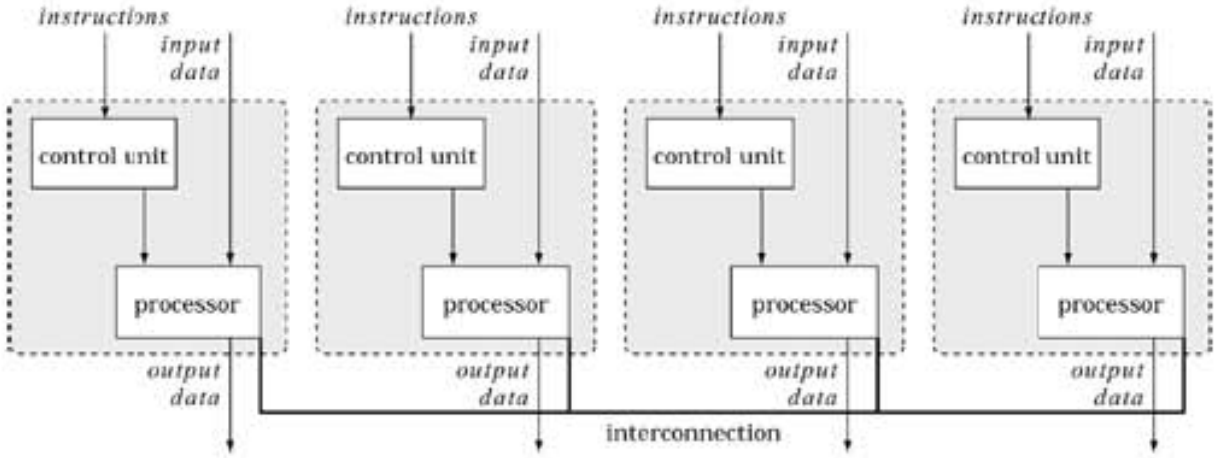
    for (i = 0; i < 1000; i++) c[i] = a[i] + b[i];
  - Diferitele iterații ale buclei sunt independente unele de altele; adică, c [0] = a [0] + b [0]; c [1] = a [1] + b [1];, etc., toate pot fi executate independent unul de celălalt.
  - Dacă există un mecanism de executare a aceleiași instrucțiuni si fiecare element de procesare poate primi datele adecvate, se poate executa această buclă mult mai rapid.

Vectorizarea nu este moarta

- In practica:
  - Calculatoarele vectoriale erau foarte frecvente în domeniul HPC, deoarece permiteau performanțe foarte mari chiar și la viteze mai mici de ceas al procesorului.
  - Până la un anumit punct au început să dispară încet.
    - Procesoarele vectoriale sunt foarte complexe și, astfel, costisitoare și funcționează slab cu probleme care nu sunt vectorizabile.
- Insa incepand cu Pentium III, Intel introduce *SSE* (Streaming SIMD Extensions), care este un set de instrucțiuni vectoriale.
  - În anumite aplicații, cum ar fi codarea video, utilizarea acestor instrucțiuni vectoriale poate oferi creșteri impresionante ale performanței.
  - Mai multe instrucțiuni vectoriale au fost adăugate cu SSE2 (Pentium 4) si SSE3 (Pentium 4 Prescott).

Multiple Instruction, Multiple Data (MIMD)

- Fiecare PE are propriul flux de instrucțiuni care operează pe propriile date.
- Este cea mai generală dintre arhitecturi: fiecare dintre celelalte cazuri pot fi mapate în arhitectura MIMD.
- Marea majoritate a sistemelor paralele moderne se încadrează în această categorie.
- Pe un computer MIMD, fiecare unitate de procesare paralelă execută operațiuni independent una de cealaltă, sub rezerva sincronizării prin trecerea mesajelor adecvate la intervale de timp specificate.
- Atât distribuția paralelă a datelor, cât și transmiterea mesajelor și sincronizarea sunt sub controlul utilizatorului.
- Ex: Intel Gamma, Delta Touchstone, Cray C-90, IBM SP2 (1990s).



SIMD si MIMD

SIMD si unitatea de control

- O singură unitate de control trimite instrucțiuni către fiecare unitate de procesare.
- Aceeași instrucțiune este executată sincron de toate unitățile de procesare.
- Example de sisteme:
  - Vechi: Illiac IV, MPP, DAP, CM-2, and MasPar MP-1 Thinking Machines CM-2 sau NCUBE Inc. computers
  - Moderne: Unități MMX în procesoare Intel și cipuri DSP, cum ar fi Sharc procesor Intel Pentium cu SSE
  - Fiecare procesor efectuează aceeași operație aritmetică (sau rămâne inactiv) în timpul fiecărui ceas al computerului, controlat de o unitate de control centrală.
- Se folosesc limbaje la nivel înalt, iar calculul și comunicarea dintre procesoare sunt sincronizate implicit la fiecare perioadă de ceas.
- Aceste îmbunătățiri arhitecturale se bazează pe natura foarte structurată (regulată) a calculelor de bază, de exemplu în procesarea imaginilor și grafică, pentru a oferi performanțe îmbunătățite.

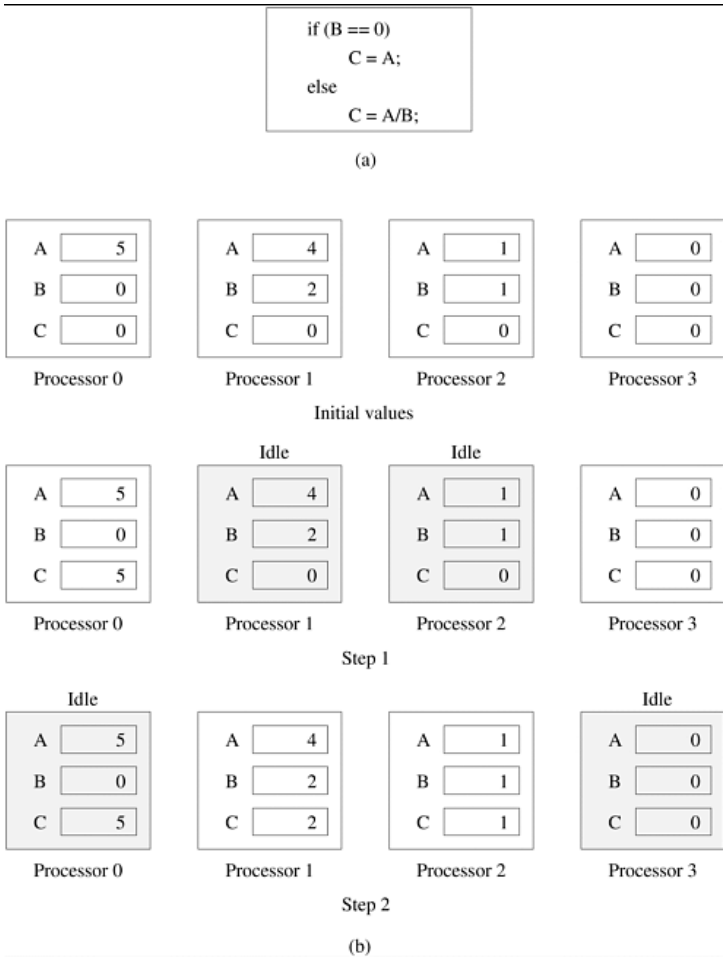
SIMD si array processors

- Dacă procesoarele SIMD sunt direcționate prin instrucțiuni emise de la o unitate de control centrală, sunt caracterizate ca ***procesoare matrice***.
  - Un număr relativ mare de procesoare relativ slabe, fiecare asociat cu o memorie relativ mică.
  - Procesoarele sunt combinate într-o topologie asemănătoare matricii, de unde și denumirea populară a acestei categorii – matrice de procesoare.
  - Gestionarea procesării are loc în unitatea de control.
  - Fiecare procesor efectuează operațiuni pe fluxuri de date separate;
  - Toate PEurile pot efectua aceeași operație sau unele pot “sari” o operație dată sau o secvență de operații.
- Comerciati pentru primele sisteme: ICL, MasPar si Thinking Machines.
- În prezent, SIMD-urile se pot intalni in cazul GPGPUrilor.
- Avantaj: procesoarele funcționează sincron, ceea ce permite urmărirea și depanarea relativ rapidă a programului.
- Dezavantaje:
  - Relativ dificil să fie folosite pentru probleme nestructurate

SIMD este o arhitectură paralelă a datelor

- Caracteristica cheie a modelului de programare este că operațiunile pot fi efectuate în paralel pe fiecare element al unei structuri de date regulate mari, cum ar fi o matrice.
- Programul este logic un singur fir de control, realizând o secvență de pași secvențiali sau paraleli.
- În timp ce conceptul SIMD funcționează bine pentru calcule structurate pe structuri de date paralele, precum matricile, de multe ori este necesar să opriți selectiv operațiunile pe anumite elemente de date.
  - Majoritatea paradigmelor de programare SIMD permit o „mască de activitate”.
    - Aceasta este o mască binară asociată cu fiecare element de date și operațiune care specifică dacă ar trebui să participe la operație sau nu.
    - Execuția condiționată poate fi în detrimentul performanței procesoarelor SIMD și, prin urmare, trebuie utilizată cu atenție.

Declaraţii condiţionale în SIMD



- Declaraţia condiţională din fig. este executată în două etape:
1. În prima etapă, toate procesoarele care au B egal cu zero execută instrucţiunea C = A. Toate celelalte procesoare sunt inactive.
2. În a doua etapă, partea „altul” a instrucţiunii (C = A / B) este executată. Procesoarele care au fost active în primul pas devin acum inactive.

- Nu este extrem de eficient!

Opțiuni de proiectare SIMD: Sincronizate vs. asincrone

- Fiecare procesor poate executa sau ignora instrucțiunea difuzată în funcție de starea sa locală sau condițiile dependente de date.
  - Acest lucru duce la o anumită ineficiență în executarea calculelor condiționate.
- De exemplu: o instrucțiune „if-then-else” este executată mai întâi activând procesoarele pentru care condiția este satisfăcută și apoi întoarcerea bitului „active” înainte de a intra în partea „else”.
  - În medie, jumătate dintre procesoare vor fi inactive pentru fiecare ramură.
- Și mai rău pentru cazul care implică ramuri cu mai multe căi.
- O posibilă solutie este utilizarea versiunii **asincrone** a SIMD,
  - Cunoscuta ca **SPMD** (spim-dee sau **single-program, multiple data**):
  - fiecare procesor rulează propria copie a programului comun.
  - avantajul este că într-un calcul „if-then-else”, fiecare procesor va petrece timp doar pe ramura relevantă.
  - dezavantajele includ nevoia de sincronizare ocazională și complexitatea mai mare a fiecărui procesor, care trebuie să aibă acum o memorie de program și o instrucțiune pentru a prelua / decoda logica.

SIMD design: Customizat vs. SIMD cu chip-uri comune

- Componente comune (off-the-shelf) :
  - componentele tind să fie ieftine din cauza producției în masă.
  - astfel de componente cu scop general vor conține probabil elemente care nu sunt necesare pentru un anumit proiect.
  - Aceste componente suplimentare pot complica proiectarea, fabricarea și testarea mașinii SIMD și pot introduce sancțiuni de viteză.
- Componente customizate:
  - inclusiv ASIC = IC-uri specifice aplicației, module multichip sau WSI = circuite integrate pe scară wafer
  - oferă în general performanțe mai bune
  - duc la un cost mult mai mare, având în vedere că costurile de dezvoltare ale acestora sunt suportate de un număr relativ mic de utilizatori de mașini paralele

Intre SIMD si MIMD

- Ineficienta SIMD => o evoluție naturală a sistemelor multiprocesoare către modelul MIMD mai flexibil
  - în special modelul de programare combinat în care există un singur program pe fiecare nod.
    - Acest model de programare îmbinat este un hibrid între modelul paralel de date și modelul de transmitere a mesajelor
- Exemplificat cu succes de Connection Machine CM-5
- In modelul SPMD (single program multiple data),
  - Programele de paraleism pe date pot activa sau dezactiva modul de transmitere a mesajelor tipic pentru MIMD,
  - ⇒ astfel se poate profita de cele mai bune caracteristici ale ambelor modele
- Alte exemple de astfel de platforme: PC-urile multiprocesoare, clusterele stațiilor de lucru, IBM SP.

SIMD pro si contra

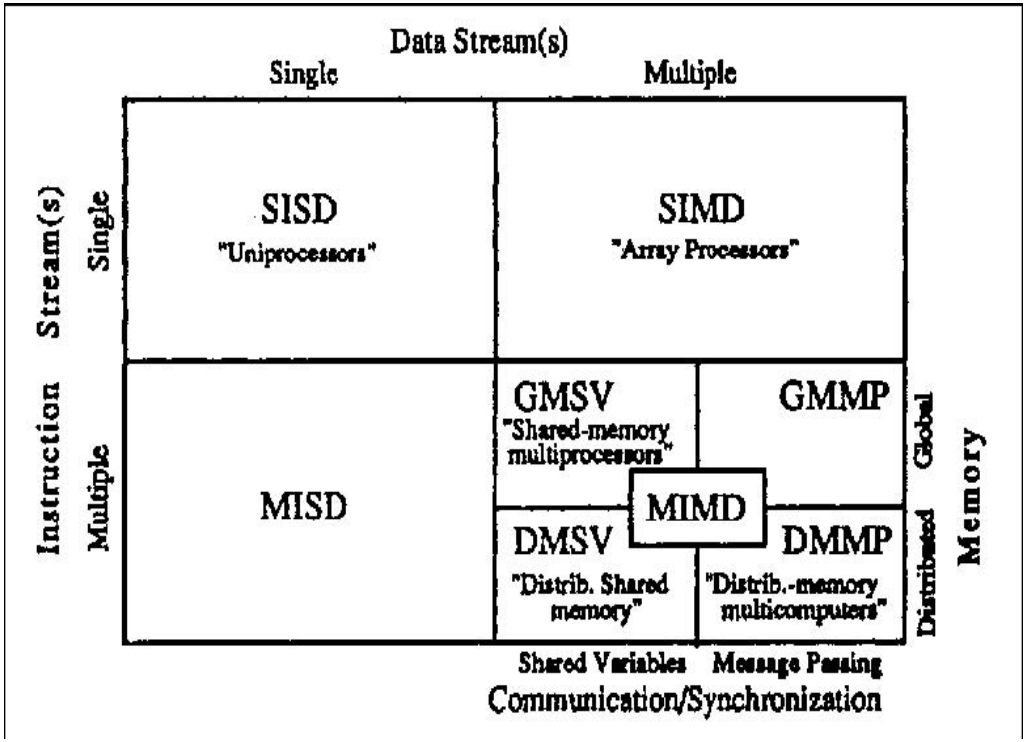
- Calculatoarele SIMD necesită mai puțin hardware decât MIMDurile
  - deoarece au o singură unitate de control globală.
  - deoarece trebuie să fie stocată doar o copie a programului.
  - Calculatoarele MIMD stochează programul și sistemul de operare la fiecare procesor.
- Popularitatea relativa a SIMDurrilor ca masini de calcul cu scop general poate fi atribuita la:
  - arhitecturile hardware specializate,
  - factori economici,
  - constrângeri de proiectare,
  - ciclul de viață al produsului,
  - caracteristicile aplicației
  - efort extins de proiectare care duce la perioade mai lungi de dezvoltare a produselor
  - natura neregulată a multor aplicații

MIMD pro si contra

- flexibilitate mai mare a arhitecturii MIMD în comparație cu SIMD
- capacitatea de a profita de microprocesoarele generale
  - evitarea ciclurilor de dezvoltare îndelungate
  - ⇒ obținerea o modalitate simpla de îmbunătățire a vitezei pentru astfel de microprocesoare.
- Cel mai eficient pentru aplicații paralele cu granulatie medie sau grosiera,
  - calculul este împărțit în sub-calcule sau sarcini relativ mari ale căror execuții sunt atribuite diferitelor procesoare.
- Printre avantajele mașinilor MIMD se numără:
  - flexibilitate în exploatarea diverselor forme de paralelism,
  - relativă ușurință de partiționare în procesoare paralele independente mai mici într-un mediu multiuser
  - expansiune mai puțin dificilă (scalabilitate).
- Dezavantajele includ:
  - comunicare interprocesoare considerabilă and
  - programare mai dificilă.

MIMD subcategorii | clasificarea Flynn–Johnson

- 1988: **E. E. Johnson** a propus o clasificare suplimentară a MIMD pe baza:
  1. structura memoriei lor: globală sau distribuită
  2. mecanism folosit pentru comunicare / sincronizare: variabile partajate sau transmiterea mesajelor.
- GMSV: **shared-memory multiprocessors**
- DMMP: **distributed-memory multicomputers**
- DMSV: uneori se numește **memorie distribuită distribuită**
  - combinați ușurința de implementare a memoriei distribuite cu ușurința de programare a schemei cu variabile partajate
- GMMP nu este utilizat pe scară largă



Memorie partajata (SM-MIMD)

- Toate procesoarele sunt conectate la o memorie comună (*RAM*).
- De obicei, toate procesoarele sunt identice și au acces egal la memorie egală
  - Sistemul se numeste **symmetric multiprocessing (SMP)**.
- Conexiunea dintre PEuri și memorie are o importanță predominantă.
- De exemplu: un sistem de memorie partajată cu conexiune **bus**.
  - Avantajului unui bus este extensibilitatea sa.
  - dezavantajul este că toate procesoarele trebuie să partajeze lățimea de bandă oferită
- Pentru a evita problema lățimii de bandă a memoriei limitate, sunt dorite conexiuni directe de la fiecare procesor la fiecare modul de memorie.
  - Acest lucru poate fi obținut folosind un **crossbar switch**.
  - Problema este complexitatea ridicată a acestora atunci când trebuie realizate multe conexiuni.
  - Această problemă poate fi slăbită prin utilizarea întrerupătorilor transversale cu mai multe etape, ceea ce la rândul său duce la timpi de comunicare mai lungi.
  - ⇒ No. CPUs&mem modulele care pot fi conectate prin comutatoare transversale este limitat.
- Avantage: toate procesoarele folosesc întregul meme
  - ⇒ Acest lucru le face ușor de programat și eficient de utilizat.
- Factorul limitant la performanța lor este numărul de procesoare și module de memorie care pot fi conectate între ele.
  - ⇒ Sistemele cu memorie partajate constau de obicei din destul de puține procesoare.

Memorie distribuita (DM-MIMD)

- Fiecare procesor are propria sa memorie locală.
- Procesoarele sunt conectate între ele.
- Cerințele impuse rețelei de comunicații sunt mai mici decât în cazul unui SM-MIMD
  - comunicarea dintre procesoare poate fi mai lentă decât comunicarea dintre procesor și memorie.
- Sistemele de memorie distribuită pot fi extrem de extinse
  - La multe mii de procesoare nu sunt neobișnuite, se numește **massively parallel processing (MPP)**.
- Pentru a utiliza efectiv performanța teoretică, este nevoie de mult mai mult efort de programare decât cu sisteme de memorie partajate.
  - Problema trebuie împărțită în părți care necesită puțină comunicare.
  - Procesoarele pot accesa numai propria lor memorie.
  - În cazul în care acestea necesită date din memoria unui alt procesor, aceste date trebuie să fie copiate.
  - Datorită rețelei de comunicații relativ lente între procesoare, aceasta trebuie evitată pe cât posibil.

ccNUMA

- Sistemele de memorie partajată suferă de o dimensiune limitată a sistemului
- Sistemele de memorie distribuită suferă de o comunicare grea între amintirile procesoarelor.
- Un compromis este arhitectura **ccNUMA (cache coerent non-uniform memory)**.
  - constă din mai multe sisteme SMP.
  - acestea sunt conectate între ele prin intermediul unei rețele de comunicații rapide, adesea întrerupătoare transversale.
  - Accesul la memoria completă, distribuită sau neunificată este posibil printr-o memorie cache comună.
  - Un sistem ccNUMA este la fel de ușor de utilizat ca un adevărat sistem de memorie partajată, în același timp este mult mai ușor de extins.
  - Pentru a obține performanțe optime, trebuie se se asigure utilizarea memoriei locale, și nu memoria celorlalte module, care este accesibilă numai prin intermediul rețelei de comunicații lente.
  - Structura modulară este un alt mare avantaj al acestei arhitecturi.
    - Majoritatea sistemului ccNUMA constau din module care pot fi conectate pentru a obține sisteme de diferite dimensiuni.

Probleme de proiectare: MPP - procesor masiv și moderat paralel

- Paralelismul masiv se refera in general la 1000 sau mai multe PEuri
- Este mai rentabil să construiți un procesor paralel dintr-un nr. relativ mic de PEuri puternice sau un număr masiv de PEuri foarte simple?
  - „efectivul de elefanți” sau „armata de furnici”?
- Nu se poate da un răspuns general la această întrebare, întrucât cea mai bună alegere depinde atât de aplicație, cât și de tehnologie.
- In anii 1980:
  - au fost construite și comercializate mai multe computere masive paralele.
- In anii 1990:
  - o schimbare generală de la paralelismul masiv la moderat (zeci la sute de procesoare),
- Noțiunea de paralelism masiv nu a fost abandonată,
  - în special la cel mai înalt nivel de performanță necesar pentru problemele Grand Challenge (vezi Top 500!)

MIMD cuplat strâns vs. slab cuplat

- Care este o abordare mai bună a HPC?
  1. folosind multiprocesoare / multicomputere special concepute
  2. o colecție de stații de lucru obișnuite care sunt interconectate la rețea generala și ale căror interacțiuni sunt coordonate de software special pentru sistem și sisteme de fișiere distribuite
     - denumită rețea de stații de lucru (NOW) sau **cluster computing**, a câștigat popularitate în ultimii ani.
- O abordare intermediară este de a lega grupuri de procesoare strâns cuplate prin intermediul rețelelor pe arii extinse:
  - Clustere de clustere = Grids
  - Aceasta este în esență o abordare ierarhică care funcționează cel mai bine atunci când există o mulțime de localități cu acces la date.

Pasarea mesajelor explicite vs. memoria partajată virtuală

- Care schemă este mai bună?
  1. obligarea utilizatorilor să specifice în mod explicit toate mesajele care trebuie trimise între procesoare
  2. permiteți-le să programeze într-un model abstract de nivel superior, cu mesajele necesare generate automat de software-ul sistemului
- Această întrebare este, în esență, foarte asemănătoare cu cea adresată în primele zile ale limbajelor de nivel înalt și memoriei virtuale:
  - La un moment dat în trecut, programarea în limbaje de asamblare și efectuarea de transferuri explicite între memoriile secundare și primare ar putea duce la o eficiență mai mare
  - În prezent, software-ul este atât de complex și compilatoarele și sistemele de operare atât de avansate încât nu mai are sens optimizarea manuală a programelor, cu excepția cazurilor limitate în timp.
- Cu toate acestea, nu suntem încă în acest moment în procesarea paralelă, iar ascunderea structurii de comunicare explicită a unei mașini paralele de programator are consecințe nedorite asupra performanței.

Multi-core

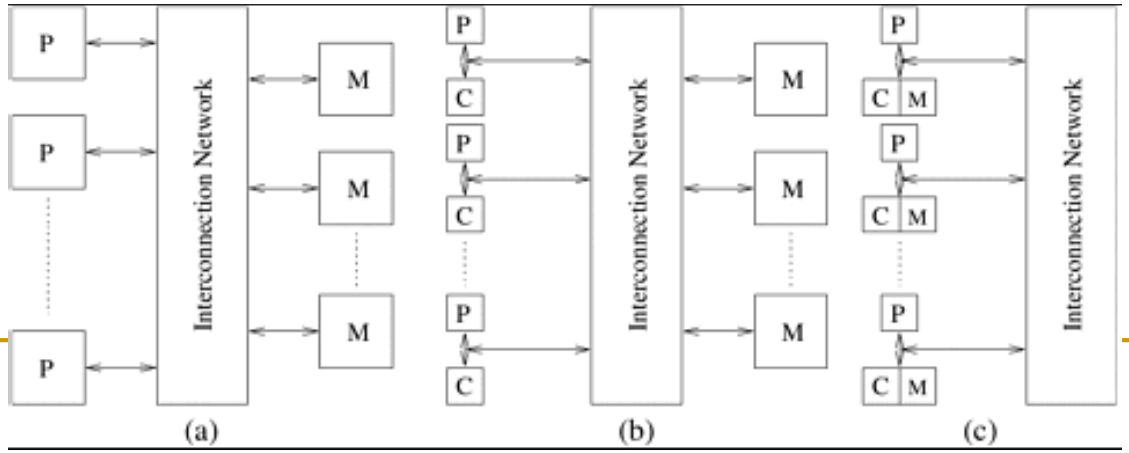
- Până în acest moment, am avut în vedere doar sisteme care procesează doar o instrucțiune pe ciclu.
  - Aceasta se aplică tuturor computerelor care conțin un singur nucleu de procesare
- Cu procesoare cu mai multe nuclee, sistemele cu un singur procesor pot avea mai multe nuclee de procesare, ceea ce le face sisteme MIMD.
- Combinarea mai multor nuclee de procesare sau procesoare (indiferent dacă scalare sau procesoare vectoriale) produce un computer care poate prelucra mai multe instrucțiuni și seturi de date pe ciclu.

Modele de comunicare pe platforme paralele

1. spațiu de date partajat
2. transmitere de mesaje

Platforma cu spațiu partajat de adresare

- Suportă un spațiu de date comun care este accesibil tuturor procesoarelor.
- Procesoarele interacționează modificând obiecte de date stocate în spațiu de adrese partajat.
- Platformele cu spații partajat de adresare care sprijină programarea SPMD sunt, de asemenea, denumite **multiprocessors**.
- Memoria în platformele cu spații de adresă partajate poate fi locală (exclusiv pentru un procesor) sau globală (comună tuturor procesatorilor).
- Dacă timpul necesar de un procesor pentru a accesa orice cuvânt de memorie din sistem (global sau local) este identic, platforma este clasificată ca un multicomputer cu **acces uniform la memorie** (**UMA**) uniform ((a) și (b))
- Dacă timpul necesar pentru accesarea anumitor cuvinte de memorie este mai lung decât pentru altele, platforma este apelată **non-uniform memory access** (NUMA) multicomputer.
- Mașini precum serverele SGI Origin 2000 și Sun Ultra HPC aparțin primelor clase de multiprocesoare NUMA.



Probleme în utilizarea adresei partajate

- Prezența unui spațiu de memorie globală face programarea unor astfel de platforme ușoară.
- Toate interacțiunile numai în citire sunt invizibile pentru programator, deoarece sunt codificate altfel decât într-un program serial.
  - Acest lucru ușurează foarte mult sarcina scrierii de programe paralele.
- Interacțiunile read/write sunt mai greu de programat decât cele de citire, deoarece aceste operațiuni necesită excluderea reciprocă pentru accesuri simultane.
  - Paradigmele de programare cu spații de adresă partajate, cum ar fi thread-urile (POSIX, NT) și directivele (OpenMP), prin urmare, susțin sincronizarea folosind semafoare și mecanisme conexe.
- Prezența cache-urilor pe procesoare ridică, de asemenea, problema copiilor multiple a unui singur cuvânt de memorie care este manipulat de două sau mai multe procesoare în același timp.
  - Suportul pentru shared-address-space presupune două sarcini majore:
    1. E nevoie de un mecanism de traducere a adreselor pentru a localiza un cuvânt de memorie
    2. Necesar că operațiunile simultane pe mai multe copii ale aceluiași cuvânt de memorie au semantică bine definite - **cache coherence mechanism**.

Probleme in utilizarea Shared-Address

- Masinile cu shared-address-space
  - acceptă doar un mecanism de traducere a adreselor
  - lasă sarcina de a asigura coerența programatorului.
    - Modelul de programare nativ pentru astfel de platforme constă în primitive precum get and put.
    - Aceste primitive permit unui procesor să obțină (și să pună) variabile stocate la un procesor la distanță.
    - Dacă una dintre copii ale acestei variabile este modificată, celelalte copii nu sunt actualizate sau invalidate automat.
- Din 2005, procesoarele compatibile x86 concepute pentru computere desktop sunt disponibile cu cel putin două „nuclee”/core (le face sisteme cu procesor dual).
  - Această putere de calcul suplimentar ieftină trebuie utilizată eficient, ceea ce necesită o programare paralelă.
  - Metodele de programare paralele care funcționează pe PC-uri cu mai multe nuclee funcționează, de asemenea, pe sisteme de memorie partajate mai mari,
  - Un program conceput pentru un cluster sau un alt tip de sistem de memorie distribuită va funcționa de asemenea bine pe PC multi-core.

Shared-address-space vs. shared-memory computers

- Termenul de calculator paralel cu memorie partajată este folosit istoric pentru arhitecturi în care memoria este partajată fizic între diferitele PEuri,
  - Fiecare procesor are acces egal la orice segment de memorie.
  - Este *identic cu modelul UMA*
- în contrast cu un computer cu memorie distribuită:
  - Diferite segmente ale memoriei sunt asociate fizic cu diferite elemente de procesare.
- Oricare dintre aceste modele **fizice**, memorie partajată sau distribuită, poate avea organizarea **logică** a unei platforme disjuncte sau ca a spațiului partajat de adresare.
  - Un sistem cu spatiu partajat de adresare si memorie distribuita este identic cu un sistem NUMA.

Platforme pentru schimb de mesaje

- Fiecare nod de procesare cu propriul său spațiu de adrese exclusiv.
- Fiecare dintre aceste noduri de procesare pot fi procesoare unice, fie un multiprocesor cu spațiu de adresare partajat
  - o tendință care câștigă rapid în calculatoarele paralele moderne care transmit mesaje.
- Instanțele unui astfel de punct de vedere provin în mod natural din stații de lucru grupate
- Pe astfel de platforme, interacțiunile dintre procesele care rulează pe noduri diferite trebuie realizate folosind mesaje (=> transmiterea mesajelor).
  - Acest schimb de mesaje este utilizat pentru a transfera date și pentru a sincroniza acțiuni între procese.
- Paradigmele care transmit mesaje acceptă executarea unui program diferit pe fiecare nod.
- Exemple dintre primele platforme paralele care acceptă paradigma de transmitere a mesajelor includ IBM SP, SGI Origin 2000, clusterele.
- Este ușor să imitați o arhitectură care transmite mesaje conținând pe un computer spațiu de adresare partajat cu același număr de noduri.
- Emularea unei arhitecturi cu spațiu de adresă partajat pe un computer care transmite mesaje este costisitoare,
  - Accesarea memoriei unui alt nod necesită trimiterea și primirea de mesaje.

Transmiterea de mesaje ca paradigmă de programare

- Interactions are accomplished by sending and receiving messages => the basic operations are send and receive.
- Since the send and receive operations must specify target addresses, there must be a mechanism to assign a unique identification or ID to each of the multiple processes executing a parallel program.
  - This ID is typically made available to the program using a function such as whoami, which returns to a calling process its ID.
- There is one other function that is typically needed to complete the basic set of message-passing operations – numprocs, which specifies the no. of processes participating in the ensemble.
- With these four basic operations, it is possible to write any message-passing program.
- Different message-passing APIs, such as the

1. Message Passing Interface (MPI) and
2. Parallel Virtual Machine (PVM),

support these basic operations and a variety of higher level functionality under different function names.

Organizarea fizica

Supercomputer

- Un **supercalculator** este printer cele mai rapide calculatoare ale timpului sau
- Supercomputerul de astăzi este desktopul sau laptopul de mâine.
- Unul dintre primii supercalculatoare cu semnificație istorică a fost Cray-1.
  - A fost utilizat cu succes în multe aplicații care implică simulări la scară largă la începutul anilor '80.
  - Cray-1 nu a fost totuși un computer paralel, dar a folosit un procesor vectorial puternic (la vremea respectivă) cu multe registre vectoriale atașate la memoria principală.
- **Astăzi, toate supercomputerele sunt computere paralele** (vezi Top500).
  - Unele se bazează pe procesoare și rețele specializate
  - Dar majoritatea se bazează pe hardware cu scop general și pe sistem de operare și software de aplicații open source.

Context istoric: intrinsec și paralelism explicit

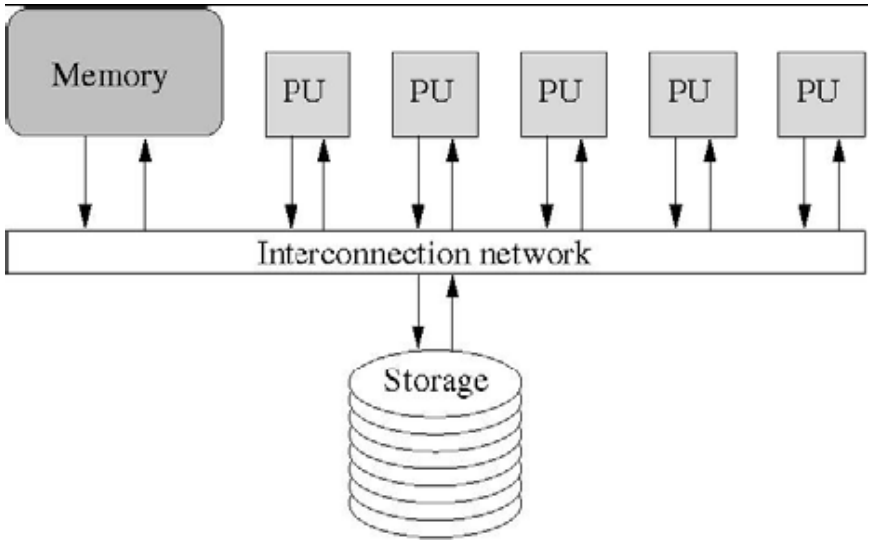
- Paralelismul a invadat inițial computerele la nivel de procesor sub mai multe aspecte:
  - Primul a avut loc în epoca procesoarelor scalare,
    - în dezvoltarea coprocesoarelor care se ocupă de unele sarcini specifice ale unității de lucru (operații matematice, comunicații, grafică, ...) și scutirea unității centrale de procesare (CPU).
  - Dezvoltarea
    1. Tehnologiei Metal-Oxide-Semiconductor (CMOS) incepand cu 1963
    2. Very-Large-Scale Integration (VLSI) incepand cu anii ‘80

  ⇒ includerea componentelor din ce în ce mai complexe în procesoare, cum ar fi pipeline și mai multe unități de calcul.
- Deoarece tehnologia CMOS este mai aproape de limitele sale fizice

⇒ paralelizarea intrinsecă a procesoarelor a fost urmată logic de apariția procesoarelor cu mai multe nuclee (multi-core).

Memorie distribuita vs. memorie partajata

Maşină paralelă cu memorie partajată



❑ Exemple de comercianti vechi:

• cele mai cunoscute exemple sunt seria Cray precum Cray-1 și 2, și Cray X-MP și Y-MP, Cray X1E;

• altii: Convex sau Alliant, Univ. of Illinois (Illiac IV), BBN, Sequent, SGI.

❑ Revenirea computerelor cu memorie partajată sub forma:

• desktop-uri multicore, care conțin de obicei de la 2 sau 16 procesoare;

• sisteme midrange cu 4-16 procesoare, utilizate în principal ca servere;

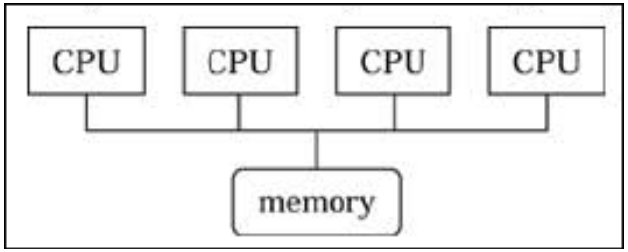
• și noduri performante pentru computerele paralele din partea de sus a liniei, care conțin de obicei 16-64 de procesoare (tehnologia de comutare este aplicată).

Avantaje si dezavantaje

- Avantaje:
  - nici nu necesită distribuții de date pe procesoare și nici mesaje cu date.
  - comunicațiile dintre PE-urile necesare pentru controlul aplicației sunt implicit efectuate prin intermediul memoriei partajate și pot fi astfel foarte rapide.
  - conexiunea de memorie este facilitată de tehnologia rapidă de autobuz sau de o varietate de tipuri de comutatoare (adică, omega, butterfly etc)
  - ușor de programat
    - paralelizarea buclelor fiind cel mai simplu și cel mai eficient mijloc de realizare a paralelismului
- Dezavantaje:
  - Memoria de memorie cache a dus la probleme relativ complicate de gestionare a datelor / coerență de cache.
  - faptul că întotdeauna a fost și este încă aproape imposibil
    - de a scala arhitecturi de memorie partajată la peste 32 de procesoare
    - de a evita simultan saturarea lățimii de bandă între procesoare și memoria globală.

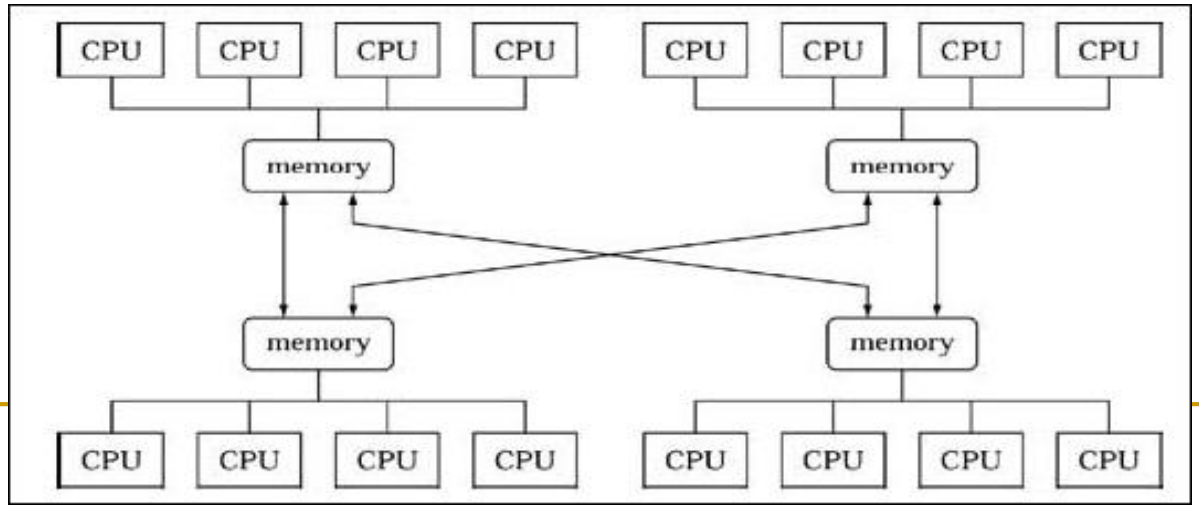
Clasa speciala 1: SMPs – symmetric multiprocessors

- toate procesoarele partajează o conexiune la o memorie comună și accesează toate locațiile de memorie la *viteze egale*.
- Sistemele SMP sunt, probabil, sistemele paralele cele mai ușoare de programat, deoarece programatorii nu trebuie să distribuie structurile de date între procesoare.
- deoarece creșterea numărului de procesoare crește competitia pentru memorie, lățimea de bandă a procesorului / memoria este de obicei un factor limitativ.
- Sistemele SMP nu se dimensionează bine și sunt limitate la un număr mic de procesoare.



Clasa speciala 2: NUMA - non-uniform memory access

- Memoria este partajata,
  - este adresabila de la toate procesoarele,
  - DAR unele blocuri de memorie pot fi asociate fizic mai strâns cu unele procesoare decât altele.
  - ⇒ Acest lucru reduce blocajul lățimii de bandă a memoriei și permite sistemelor cu mai multe proc;
  - ⇒ Timpul de acces de la un procesor la o locație de memorie poate fi semnificativ diferit în funcție de cât de „aproape” este locația de memorie a procesorului
- Pentru a atenua efectele accesului neuniform, fiecare procesor are o memorie cache, împreună cu un protocol pentru menținerea coerentă a intrărilor din cache
  - un alt nume pentru aceste arhitecturi este **cache-coherent non-uniform memory access systems** (ccNUMA)



Dezavantaje ale centralizării memoriei & solutii

- implică o lățime de bandă de memorie foarte mare, potențial cu concurență, pentru a evita blocajele
  - ⇒ rețeaua de interconectare dintre memorie și PE, precum și viteza controlerului de memorie sunt adesea factorii limitatori ai nr. de PE-uri incluse în acest tip de mașină.
- Solutii?
  1. Procesoarele pot accesa memoria printr-o rețea specială de procesare la memorie, care trebuie să aibă o **latență foarte mică**
     - o provocare destul de dificilă pentru mai multe decât câteva procesoare
  2. trebuie utilizate tehnici de ***ascundere a latenței în memorie***
     - Un exemplu de astfel de metode este utilizarea multitreading în procesoare, astfel încât acestea să continue cu funcții de procesare utile în timp ce așteaptă ca cererile de acces în memorie în așteptare să fie deservite.
  3. [optional] rețea procesor-procesor poate fi utilizată în scopuri de coordonare și sincronizare.

Exemple de rețele de la procesor la memorie și de la procesor la procesor

- Presupunem:
  - *p* elemente de procesare,
  - *m* module de memorie

1. Crossbar switch: complexitate O(*pm*), scumpe pentru un numar mare de p-uri
2. Unul sau mai multe magistrale (ultimele cu conectivitate completa sau partiala)
3. Retele de interconectare multietape (MIN); mai ieftine decat 1 si cu banda mai lata decat 2

Cache

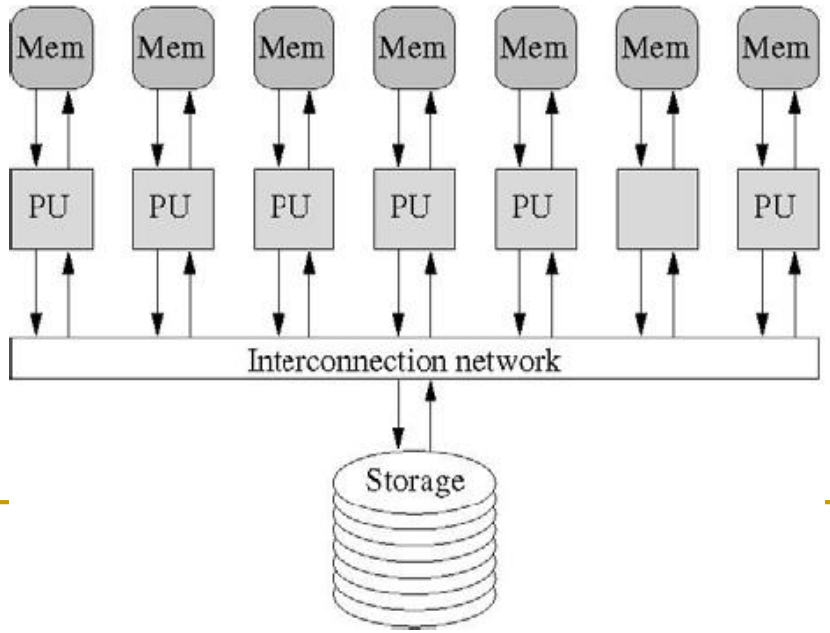
- Motivație: reducerea cantitatii de date care trebuie să treacă prin rețeaua de interconectare procesor-la-memorie
- Utilizarea unei memorii cache private de dimensiuni rezonabile în fiecare procesor.
- Motivul pentru care folosirea memoriilor cache reduce traficul prin rețea este același ca și pentru procesoarele convenționale:
  - localitatea accesului la date,
  - acces repetat la aceleași date,
  - eficiența mai mare a blocului, spre deosebire de transferul de date cu cuvânt.
- Mai multe memorii cache generează ***problema coerenței cache-ului***:
  - Copiile de date din memoria principală și din diverse cache-uri pot deveni inconsistente.
  - Abordare:
    1. A nu se memora în cache datele partajate+ a nu permite o singură copie cache.
       - Dacă volumul de date partajate este mic și accesul la acestea sunt rare, aceste politici funcționează destul de bine.
    2. A nu se pune in cache date partajate „care se pot scrie” + a nu permite o singură copie cache.
       - Datele partajate numai în citire pot fi plasate în mai multe cache-uri fără nici o complicație.

Protocol de coerență în cache

- Introduce un surplous netrivial, dependent de protocolul de coerență utilizat, dar elimină restricțiile anterioare
- Exemple:
  - *snoopy cache pentru sisteme bazate pe magistrale*
    - fiecare cache monitorizează toate transferurile de date din autobuz pentru a vedea dacă va fi afectată validitatea datelor pe care le deține
  - *scheme bazate pe directoare*
    - unde datele partajate care pot fi scrise sunt „deținute” de un singur procesor sau cache la un moment dat, cu un director folosit pentru a determina locațiile fizice

Masini paralele cu memorie distribuita

- PE-urile sunt încă legate între ele printr-o rețea de interconectare
- fiecare PU are propria sa memorie cu un acces exclusiv.
- unele resurse care sunt încă partajate, cum ar fi, de exemplu, stocarea în masă sau operațiunile de I / O
- Primele versiuni comercializate de: Intel, NCube, Inmos, Convex, Cray, IBM.
- Masine vechi reprezentative: Cray T3D/T3E, Intel iPSC/2 and Paragon, Connection Machines (CM-1 to CM-5), MasPar (MP1& MP2).
- Recente: IBM Blue Gene sau Cray Red Storm.



Avantaje & dezavantaje

- Avantaje:
  - pot fi scalate la un număr mare de procesoare.
    - Chiar și unele dintre mașinile timpurii cu memorie distribuita au folosit cu succes până la 1024 de procesoare.
  - o mai mare flexibilitate în ceea ce privește proiectarea și configurația le-a făcut mai viabile din punct de vedere financiar decât predecesorii lor
- Dezavantaje:
  - mai dificil de scris și depanat decât memoria partajată
  - necesitatea unei distribuții de date în memoria procesoarelor
  - utilizarea mesajelor care trec între procesoare pentru schimbul de date sau informații pentru controlul aplicației
  - performanțele rețelei de interconectare între PE-uri sunt, de asemenea, un punct critic

Rețele de interconectare

- Fiecare procesor este de obicei conectat la rețea prin mai multe linkuri sau *canale*
- *Rețelele directe:* canalele procesorului sunt conectate direct la omologii lor în alte procesoare, conform unor tipuri de interconectare sau topologie.
- Exemple:
  1. **Bus**: o rețea cu congestie, dar ieftină și scalabilă în ceea ce privește costurile rețelei.
  2. **Rețelele dinamice:** de exemplu, un *crossbar* care este greu de scalat și care are foarte multe comutatoare, este scumpă și de obicei este utilizată doar pentru un număr limitat de procesoare.
  3. **Rețelele statice**: de exemplu, retele Ω, matrici, inele, plase, tor, hipercub.
  4. **Combinatii**: de examplu, o magistrala care conectează clustere care sunt conectate printr-o rețea statică.

Clasificare: MPP si clustere

- MPP (procesoare masiv paralele)
  - procesoarele și infrastructura de rețea sunt strâns cuplate și specializate pentru a fi utilizate într-un computer paralel.
  - extrem de scalabil, în unele cazuri utilizand multe mii de procesoare într-un singur sistem.
- Clustere:
  - PE-uri de ordinal zecilor

NUMA si sisteme hibride

- Mașinile MIMD cu memorie distribuită sunt uneori descrise ca arhitecturi de acces la memorie nonuniforme (NUMA).
  - arhitectură cu memorie all-cache sau cache-only (COMA) pentru astfel de masini.
- Hibrid:
  - grupuri de noduri cu spații de adrese separate în care fiecare nod conține mai multe procesoare care partajează memoria.
  - fabricate din grupuri de SMP conectate printr-o rețea rapidă sunt în prezent tendința dominantă în calculul de înaltă performanță.
    - Incepand cu 2003, cele mai rapide calculatoare din lume (top 500) sunt sisteme hibride.