Curs 2

Programare Paralela si Distribuita

- Arhitecturi paralele
- Clasificarea sistemelor paralele
- *Cache Consistency*
- Top 500 Benchmarking

Clasificarea sistemelor paralele -criterii-

Resurse

- numărul de procesoare şi puterea procesorului individual;
- Tipul procesoarelor – omogene- heterogene
- Dimensiunea memoriei

Accesul la date, comunicatie si sincronizare

- complexitatea reţelei de conectare şi flexibilitatea sistemului
- distribuţia controlului sistemului,
  - dacă multimea de procesoare este coordonata de catre un procesor sau
  - dacă fiecare procesor are propriul său controller;
- Modalitatea de comunicare (de transmitere a datelor);
- Primitive de cooperare (abstractizari)

Performanta si scalabilitate

- Ce performanta se poate obtine?
- Ce scalabilitate permite?

Clasificarea Flynn

Michael J. Flynn în 1966

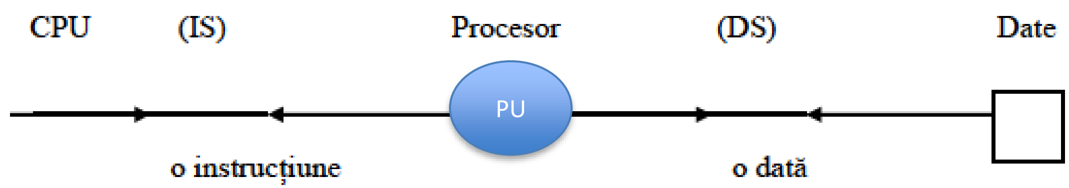
- SISD: sistem cu un singur flux de instrucţiuni şi un singur flux de date;
- SIMD: sistem cu un singur flux de instrucţiuni şi mai multe fluxuri de date;
- MISD: sistem cu mai multe fluxuri de instrucţiuni şi un singur flux de date;
- MIMD: cu mai multe fluxuri de instrucţiuni şi mai multe fluxuri de date.

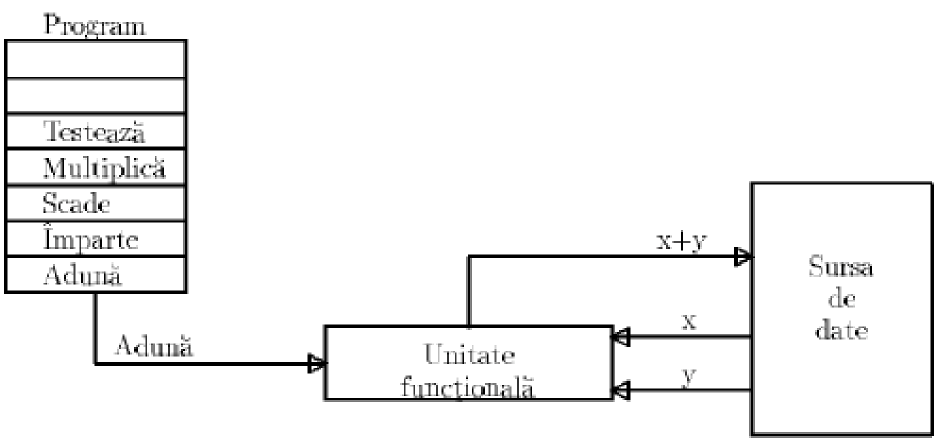
(imagini urm. preluate din ELENA NECHITA, CERASELA CRIŞAN, MIHAI TALMACIU, ALGORITMI PARALELI SI DISTRIBUIŢI)

SISD(Single instruction stream, single data stream)

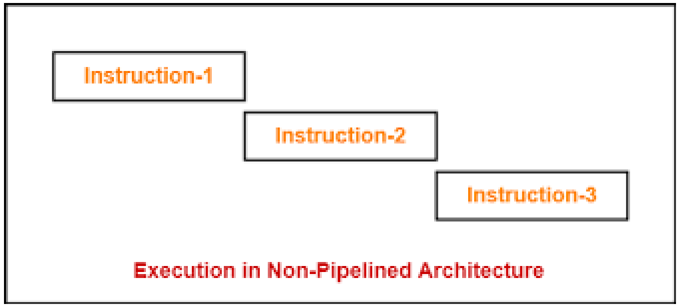
Flux de instrucțiuni singular, flux de date singular (SISD)-

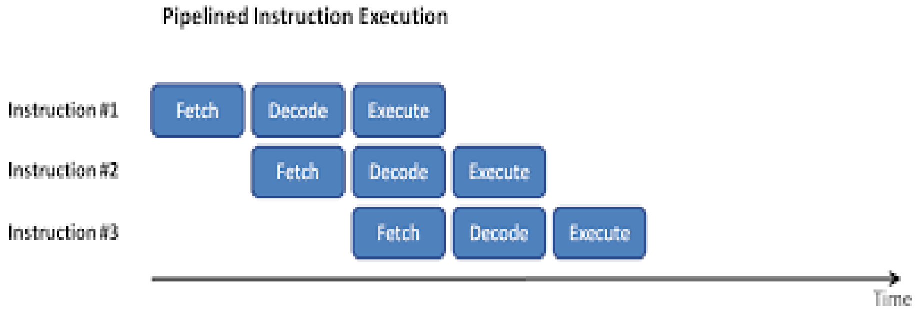
- microprocesoarele clasice cu arhitecturi von Neumann
- Functionare ciclica: preluare instr., stocare rez. in mem. , etc.





Non-pipelined vs. pipeline procesors



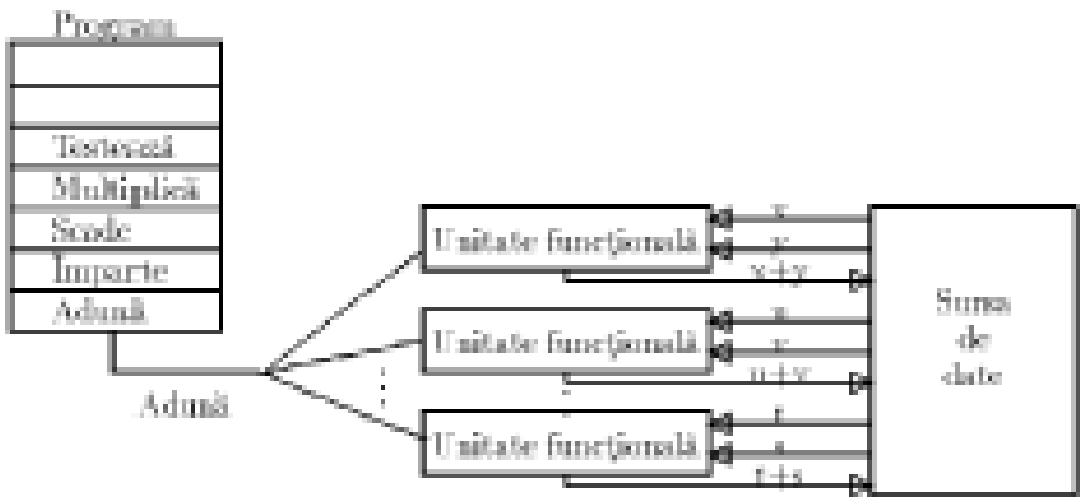


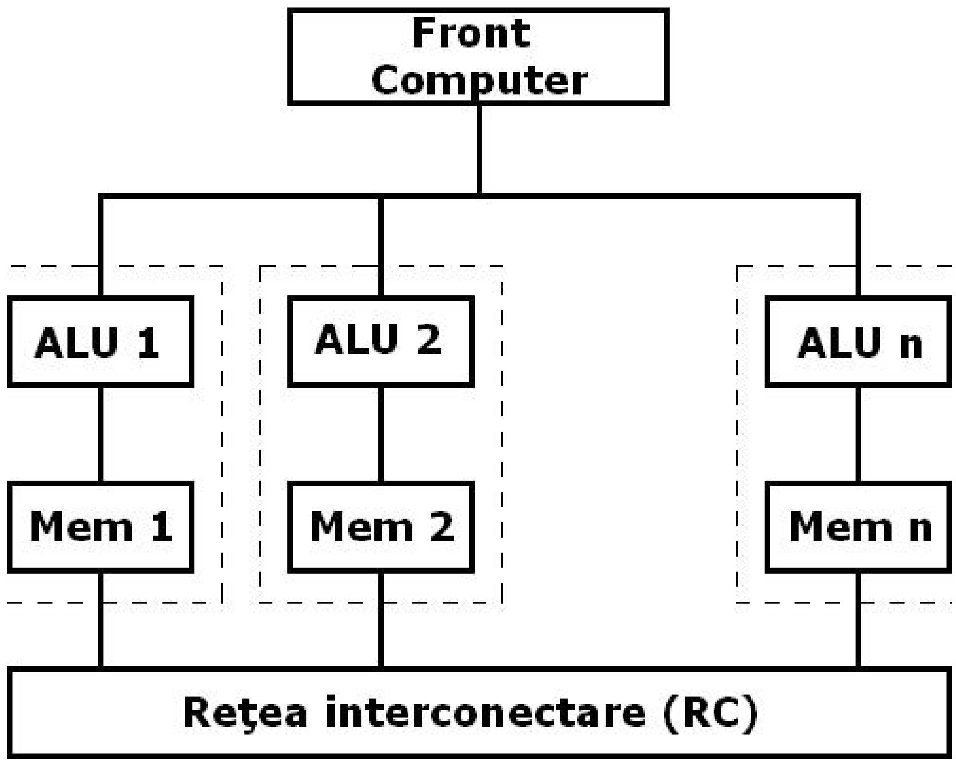
Pipeline Execution

Fetch	add				
Decode		add			
Access			add		
Execute				add	
Store					add
Time →	1	2	3	4	5

SIMD (Single instruction stream, multiple data stream)

Flux de instrucțiuni singular, flux de date multiplu

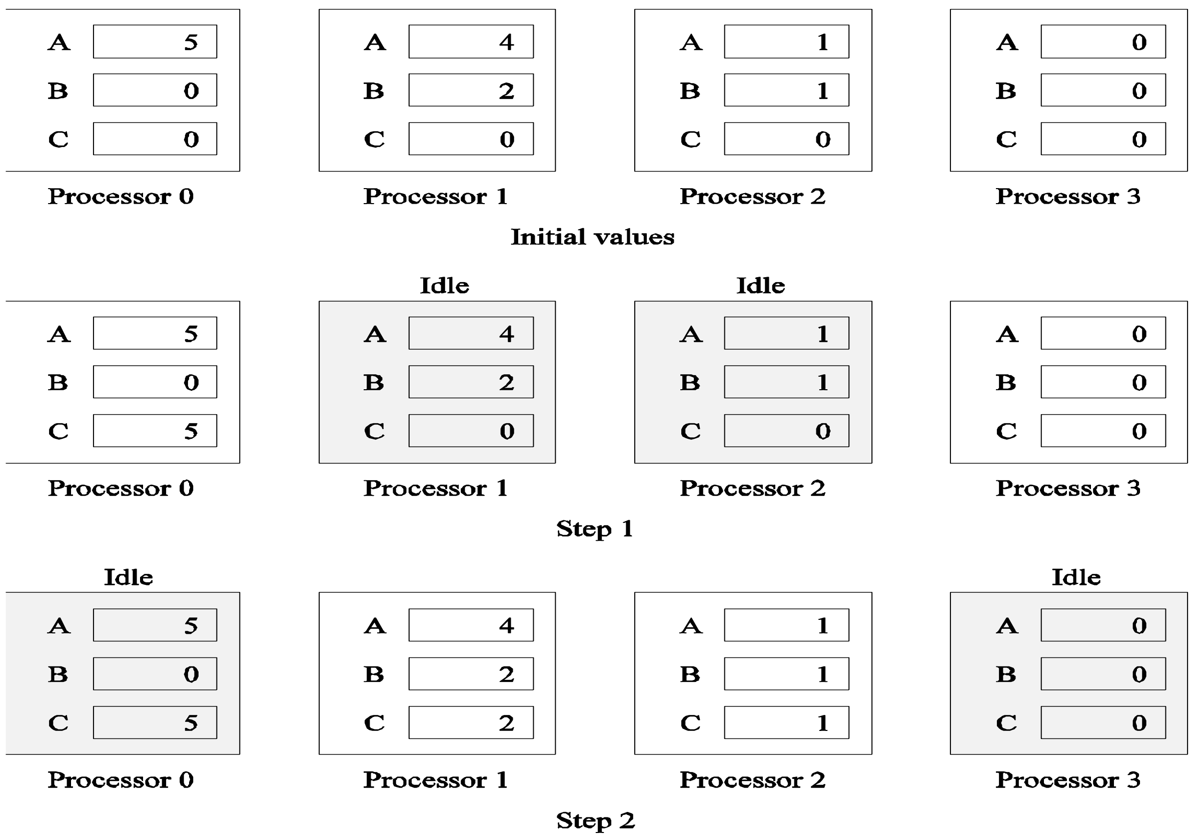



Front Computer
ALU 1
ALU 2
ALU n
Mem 1
Mem 2
Mem n
Reţea interconectare (RC)

Executie conditionala in SIMD Processors

```
if (B == 0)
        C = A;
else
        C = A/B;
```

(a)

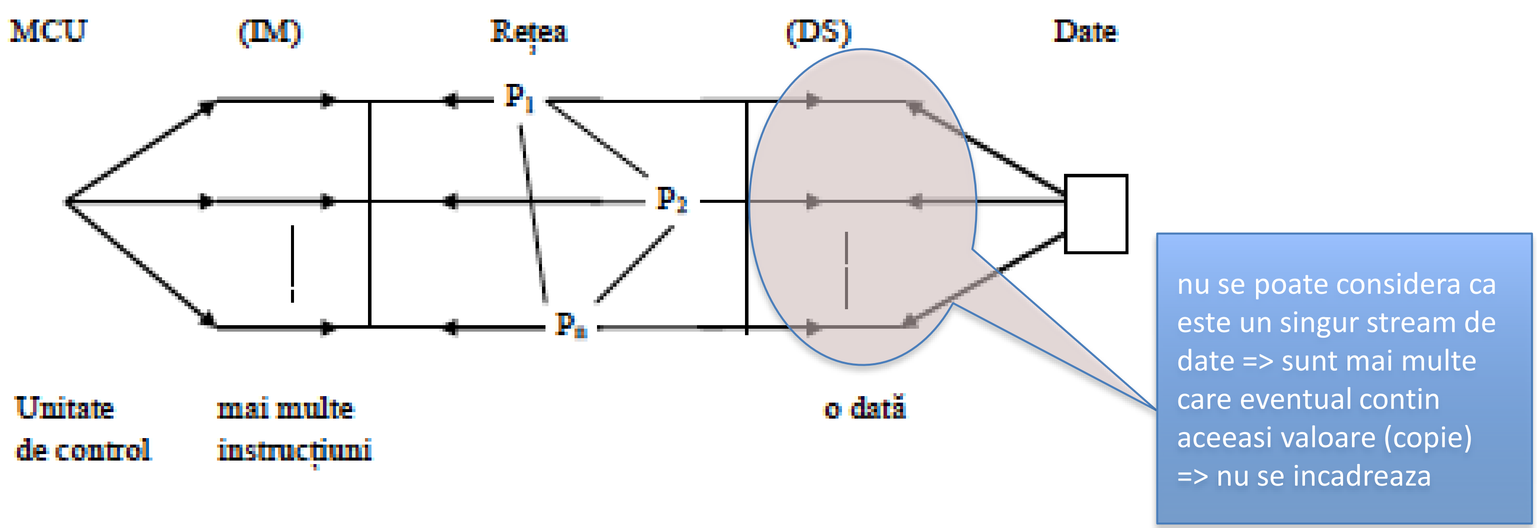


(b)

MISD (multiple instruction stream, single data stream)

Flux de instrucțiuni multiplu, flux de date singular

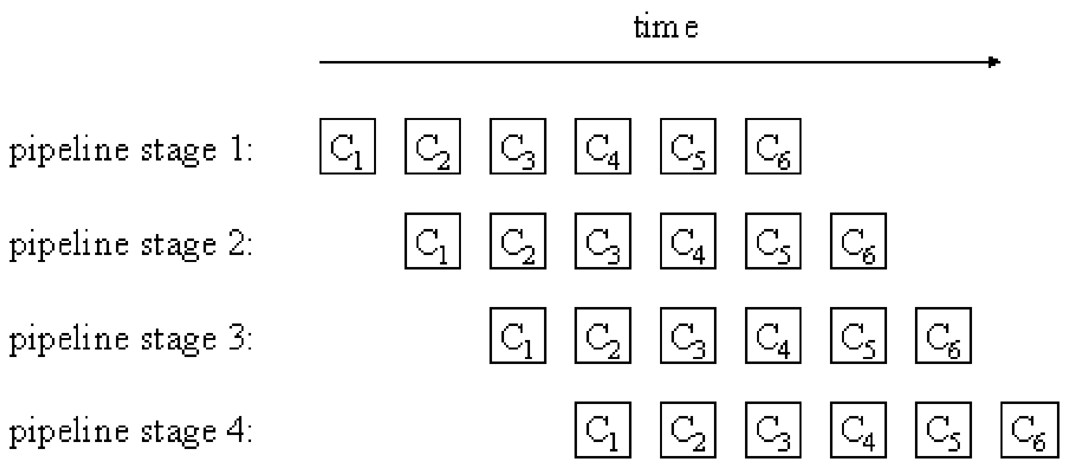
- multime vida !!!



~~~ procesoare pipeline:

intr-un **procesor pipeline** exista un singur flux (stream) de date dar aceasta trece prin transformari succesive (mai multe instructiuni) iar paralelismul este realizat prin execuţia simultana a diferitelor etape de calcul asupra unor date diferite (secventa de date care intra succesiv pe streamul de date)
~~~

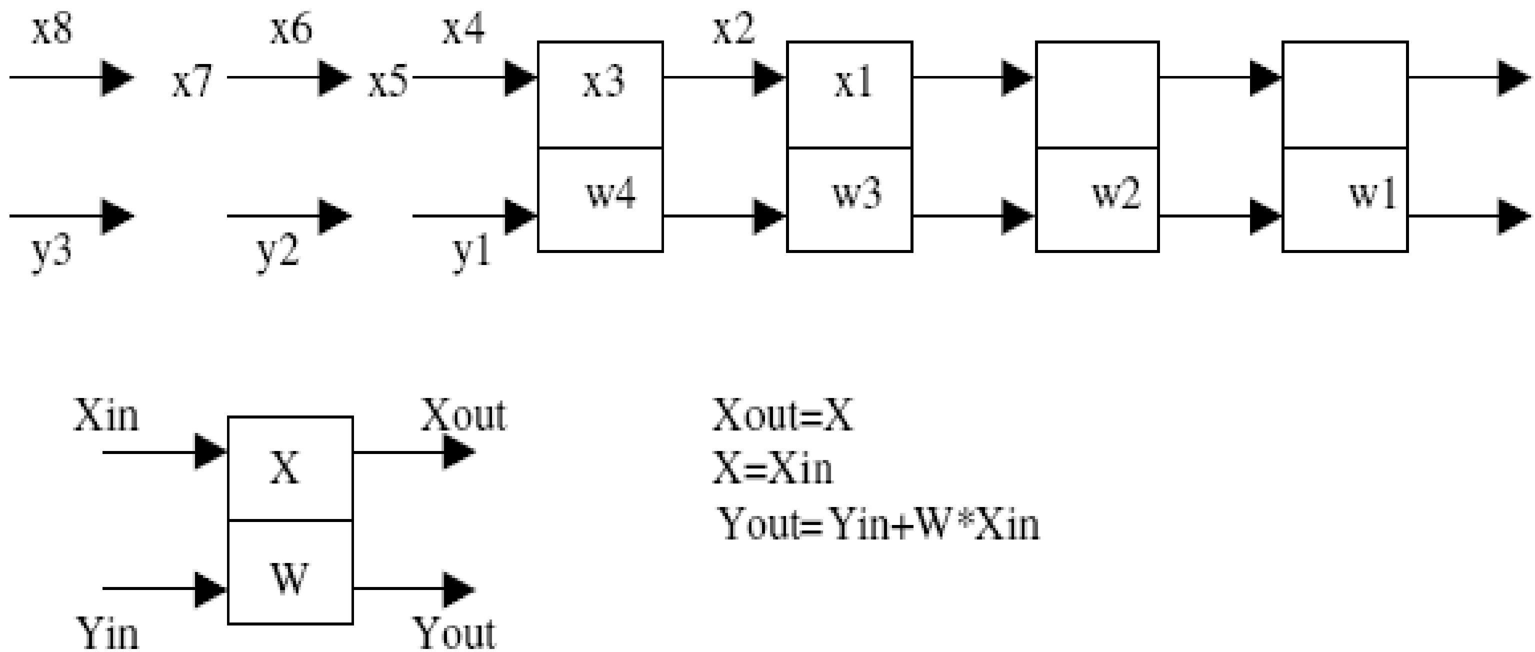
Pipeline computation



Exemplu – retea liniara (pipeline)

Exemplu: se consideră un sistem simplu pentru calcularea convoluţiilor liniare, utilizând o reţea liniară de elemente de prelucrare:

$$y(i) = w1*x(i)+ w2*x(i+1)+w3*x(i+2)+w4*x(i+3)$$



Reţea liniară pentru calcularea convoluţiilor liniare.

Arhitectura sistolica

Orchestrate data flow for high throughput with less memory access

Different from pipelining

Nonlinear array structure, multidirection data flow, each PE may have (small) local instruction and data memory

Different from SIMD

Each PE may do something different

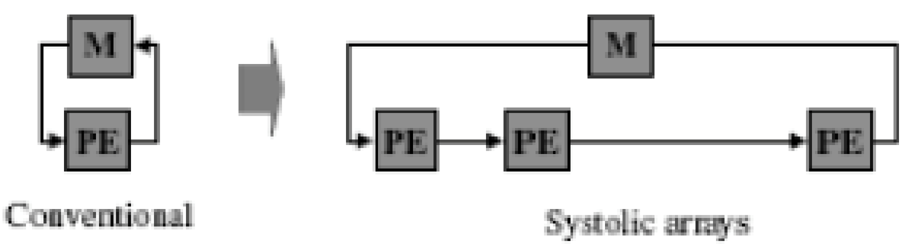
Initial motivation

VLSI enables inexpensive special-purpose chips

Represent algorithms directly by chips connected in regular pattern

Very-large-scale integration

Systolic Architectures



Replace a processing element(PE) with an array of PE's without increasing I/O bandwidth

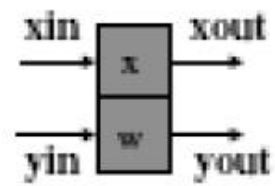
Exemplu: matrix-vector multiplication

$$y_i = \sum_{j=1}^{n} a_{ij} x_j, i = 1, \ldots, n$$

Recursive algorithm

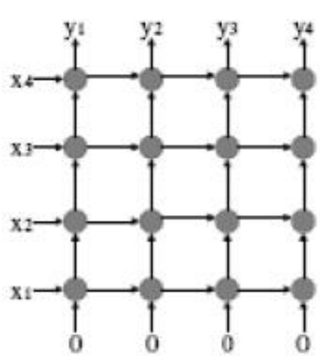
```
for i = 1 to n
  y(i,0) = 0
  for j = 0 to n
    y(i,0) = y(i,0) + a(i,j) * x(j,0)
```

Use the following PE



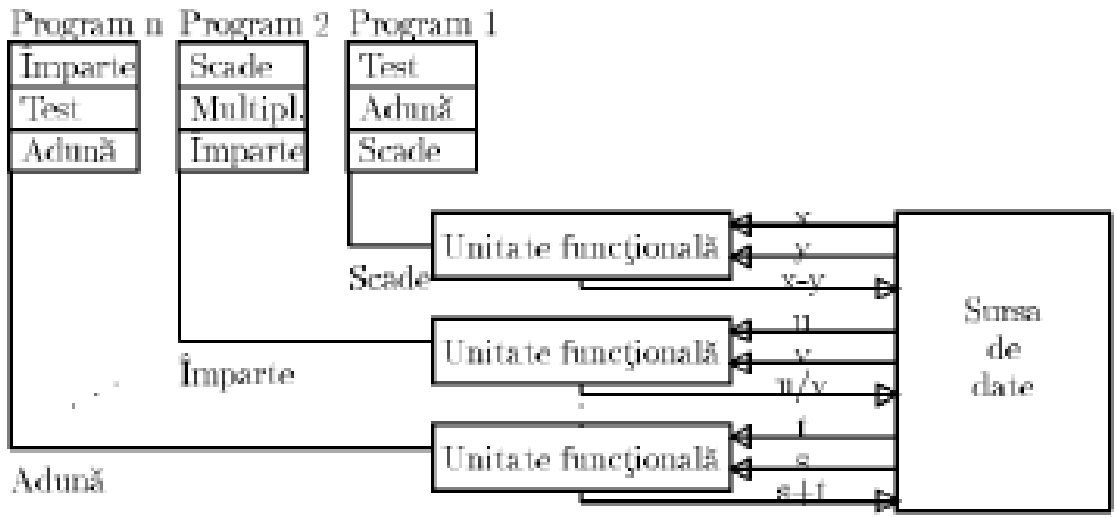
xout = x
x = xin
yout = yin + w * xin

Systolic Array Representation of Matrix Multiplication

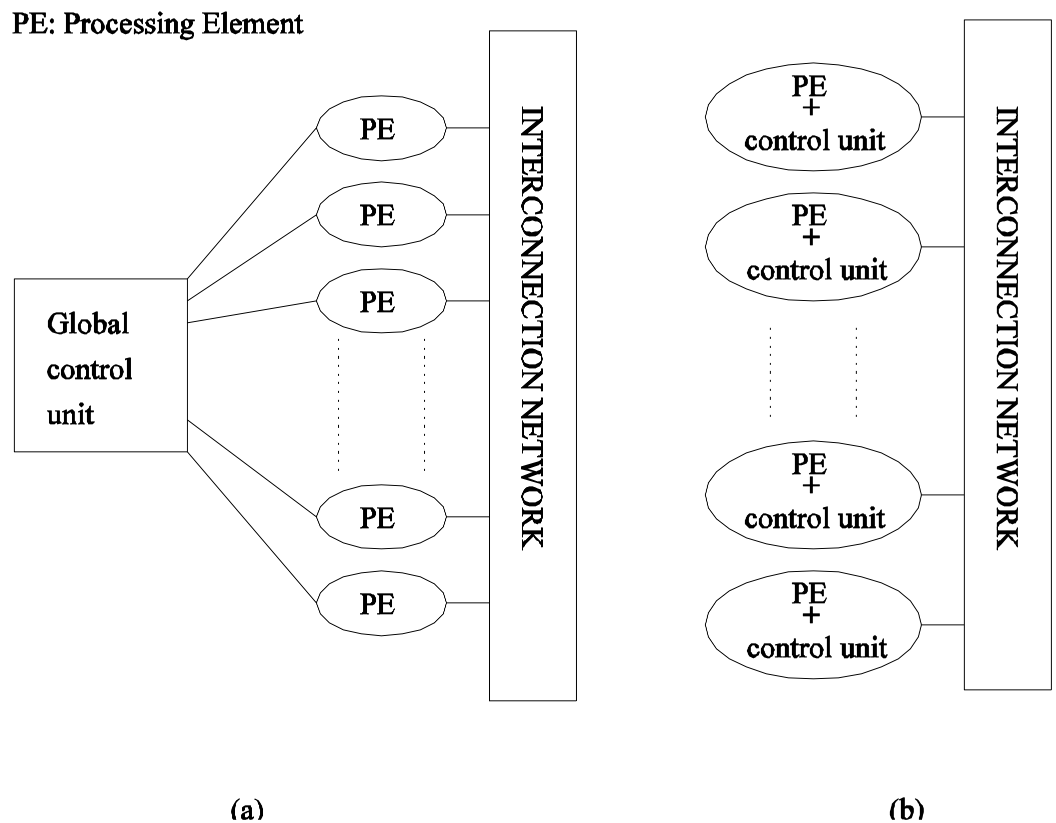


MIMD (multiple instruction stream, multiple data stream)

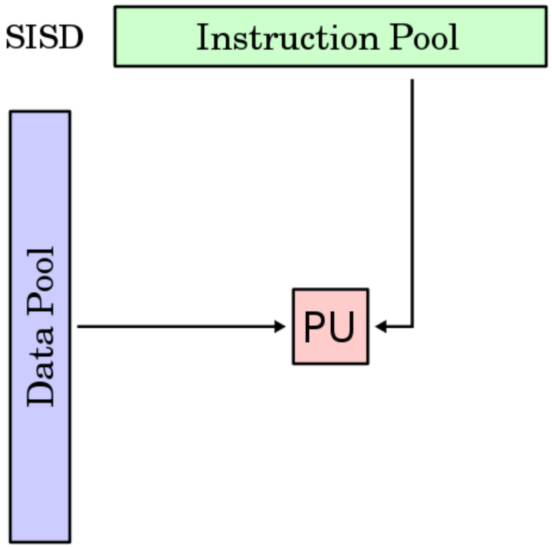
Flux de instrucțiuni multiplu, flux de date multiplu

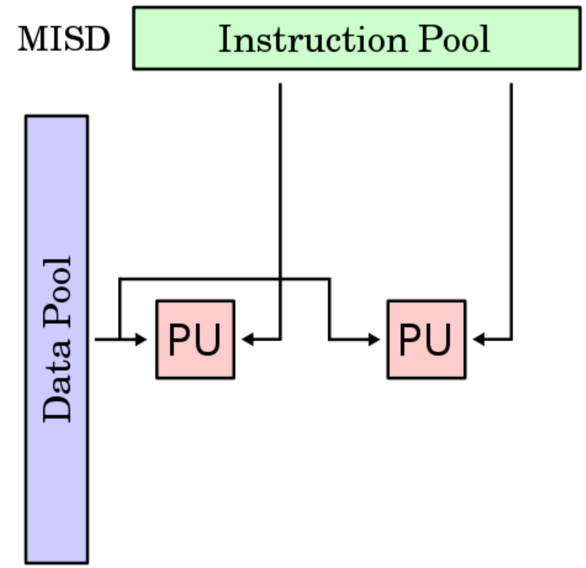


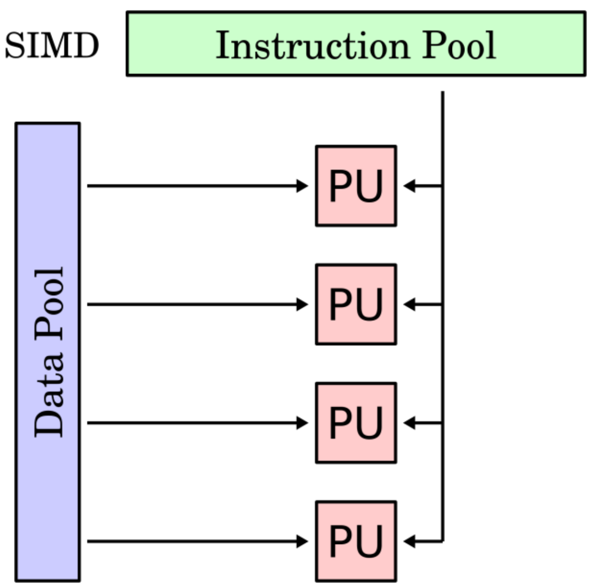
SIMD versus MIMD

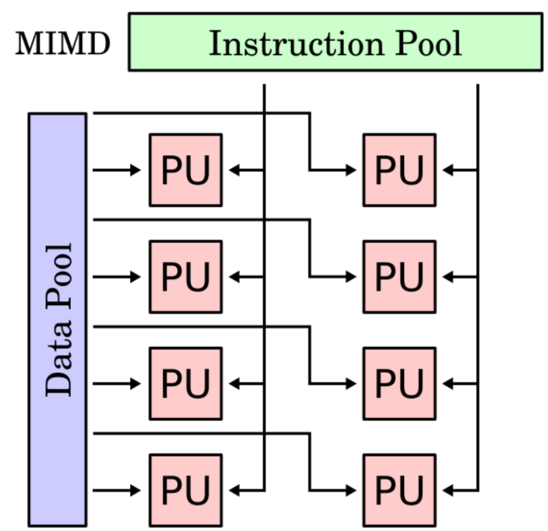


Sumar -scheme Comparative – clasificare Flynn





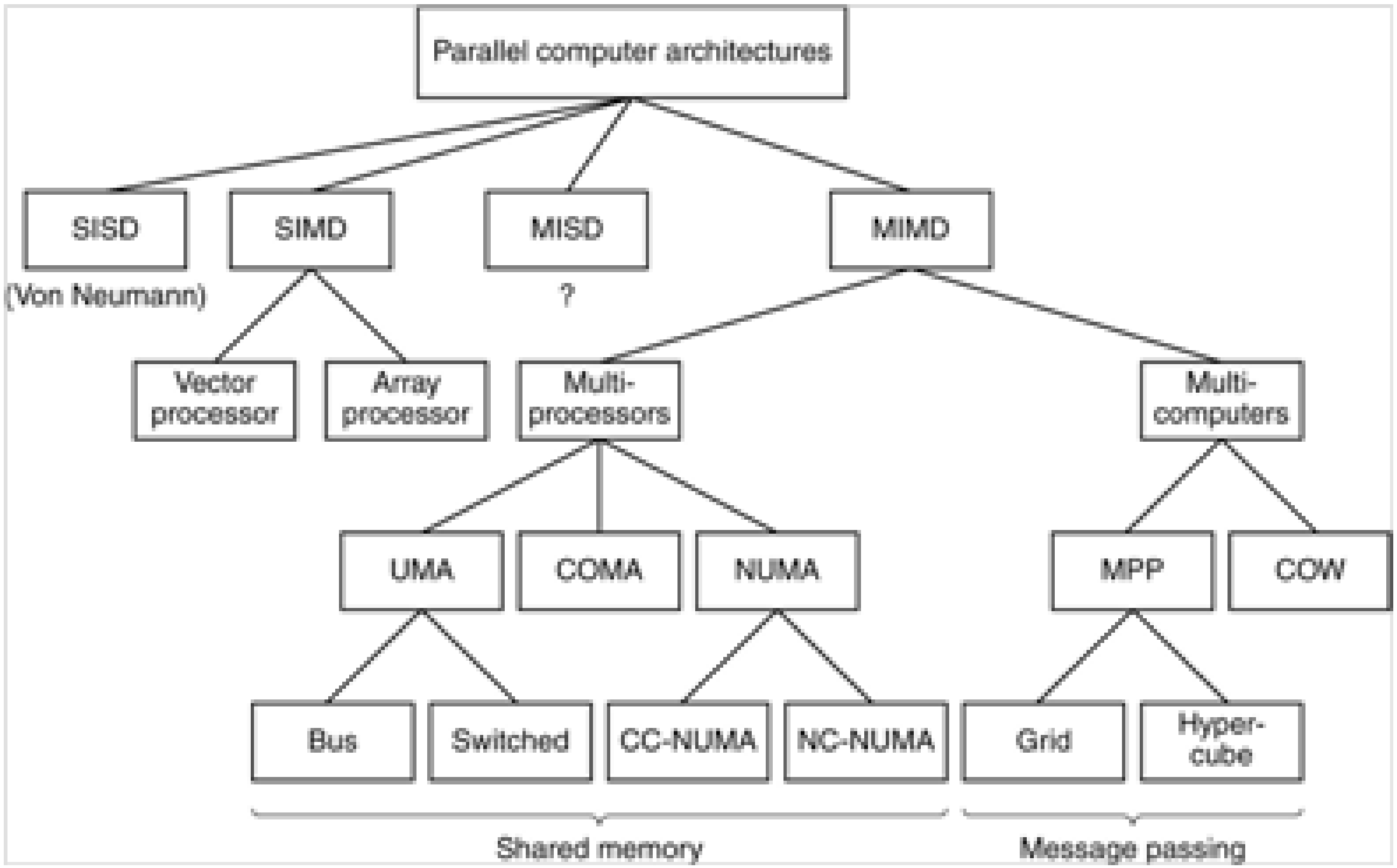




Paralelizare la nivel hardware – istoric

- Etapa 1 (1950s): executie secventiala a instructiunilor
- Etapa 2 (1960s): *sequential instruction issue*
  - Executie Pipeline,
  - *Instruction Level Parallelism* (ILP)
- Etapa 3 (1970s): procesoare vectoriale
  - Unitati aritmetice care fol. Pipeline
  - Registrii, sisteme de memorie paralele *multi-bank*
- Etapa 4 (1980s): SIMD si SMPs
- Etapa 5 (1990s): MPPs si clustere
  - *Communicating sequential processors*
- Etapa 6 (>2000): many-cores, multi-cores, acceleratori, heterogenous clusters

Vedere generala



MIMD

- Clasificare in functie de tipul de memorie
  - partajata
  - distribuita
  - hibrida

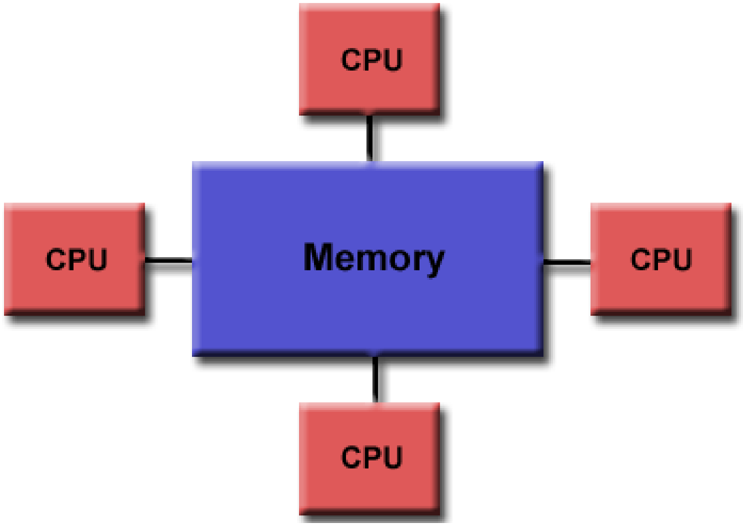
Memorie partajata/ Shared Memory

- Toate procesoarele pot accesa intreaga memorie -> un singur spatiu de memorie (*global address space.)*

- Shared memory=> 2 clase mari: ***UMA*** and ***NUMA***.

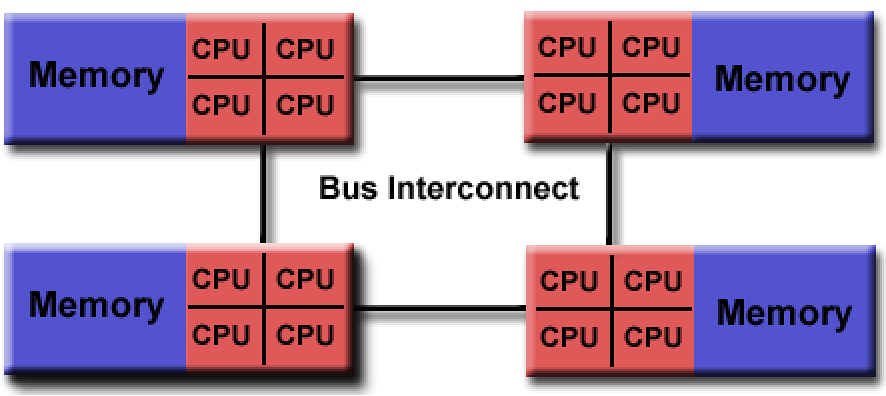
Shared Memory (UMA)

- **Uniform Memory Access (UMA):**
- Acelasi timp de acces la memorie
- **CC-UMA** - Cache Coherent UMA. (daca un procesor modifica o locatie de memorie toate celelalte “stiu” despre aceasta modificare.
  Cache coherency se obtine la nivel hardware.



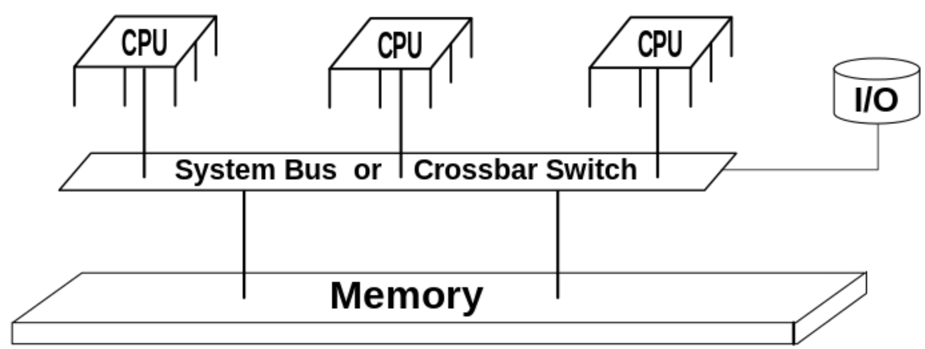
Non-Uniform Memory Access (NUMA):

- Se obtine deseori prin unirea a 2 sau mai multe arhitecturi UMA
- Nu e acelasi timp de acces la orice locatie de memorie
- ***Poate*** fi si varianta CC-NUMA - Cache Coherent NUMA
  - ex. HP's Superdome, SUN15K, IBMp690

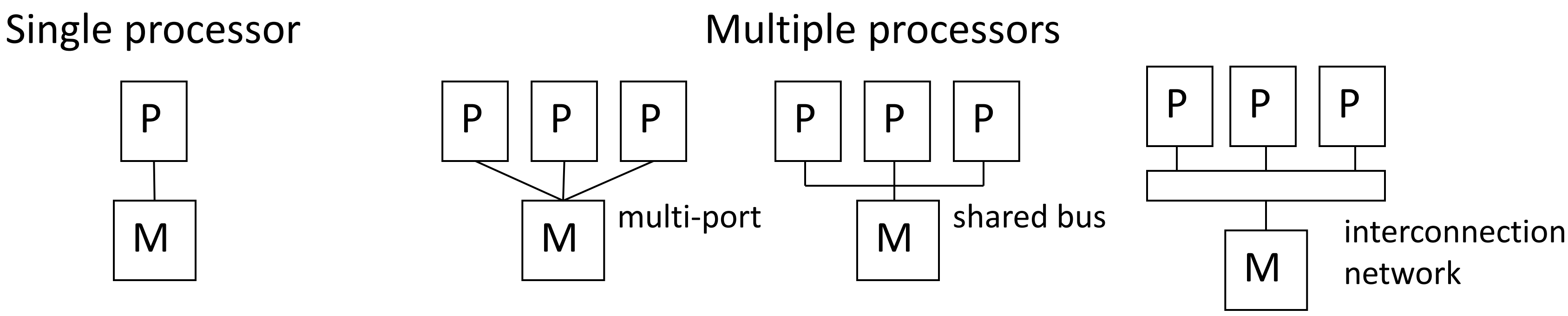


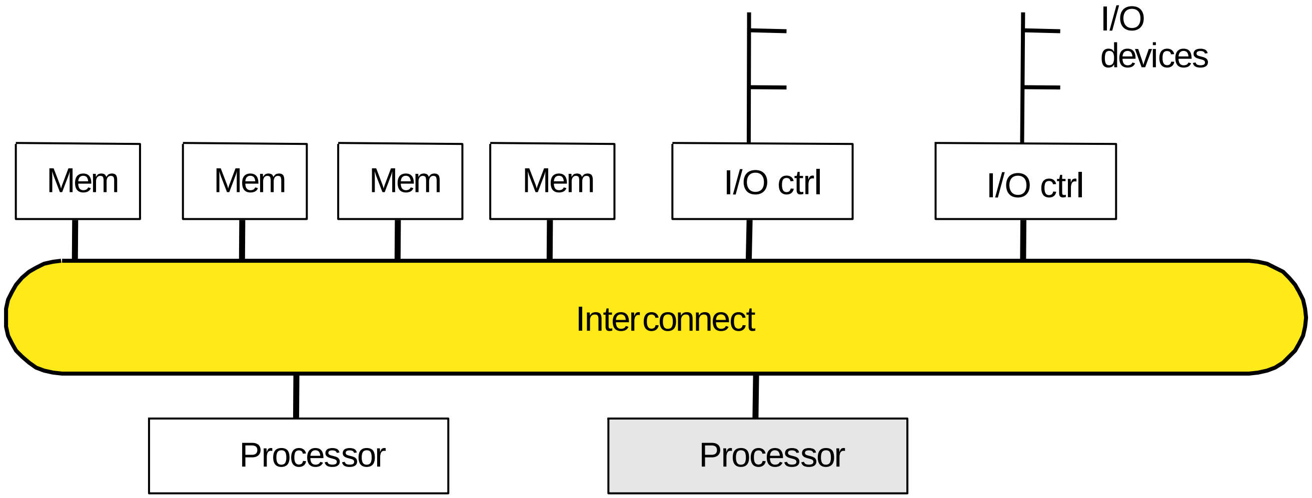
SMP Symmetric multiprocessor computer

- acces similar la toate procesoarele dar si la I/O devices, USB ports ,hard disks,...
- o singura memorie comuna
- un sistem de operare
- controlul procesoarelor – egal (similar)
- distributia threadurilor – echilibrata+echidistanta
- exemplu simplu: 2 procesoare Intel Xeon-E5 processors ->aceeasi motherboard
- Ex. - servere



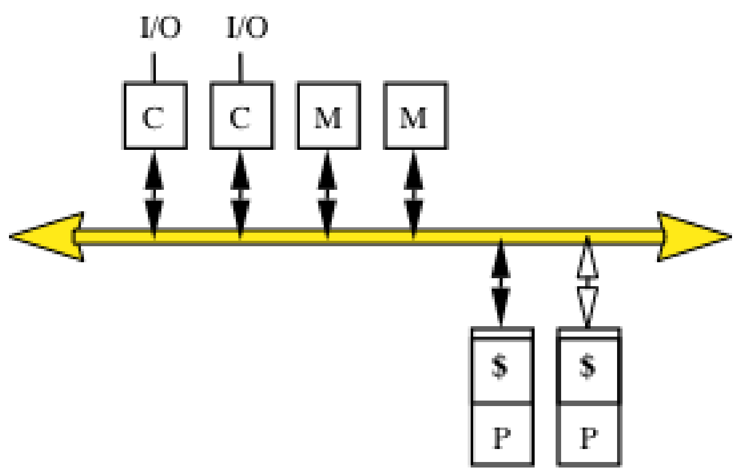
Shared Memory Multiprocessors (SMP) - overview



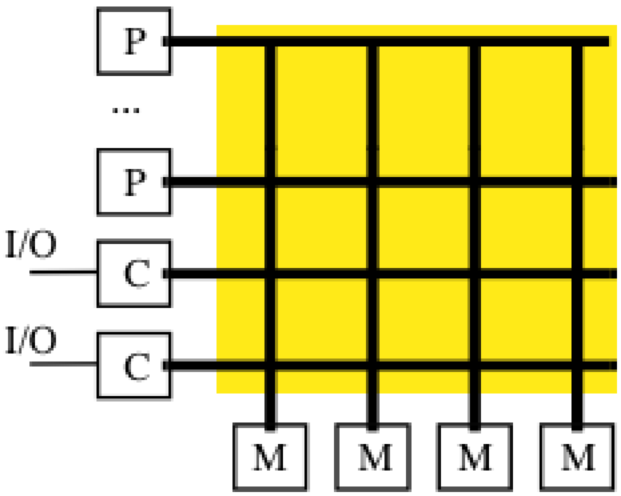


Bus-based SMP(Symmetric Multi-Processor)

- *Uniform Memory Access* (*UMA*)
- Pot avea module multiple de memorie



Crossbar SMP



Parametrii de performanta corespunzatori accesului la memorie

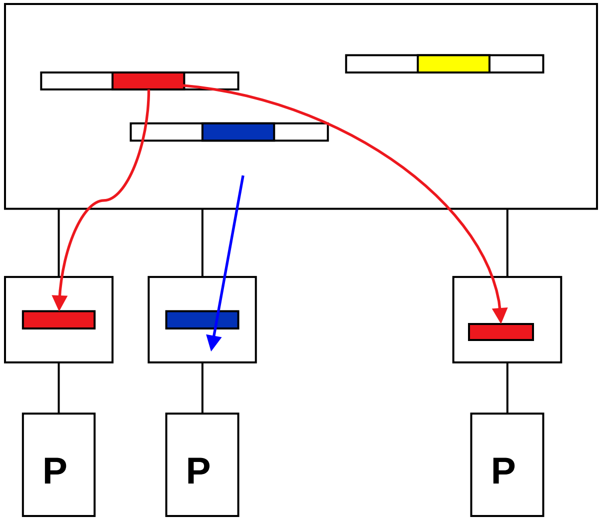
- Latenta = timpul in care o data ajunge sa fie disponibila la procesor dupa ce s-a initiat cererea.

- Largimea de banda (*Bandwidth*) = rata de transfer a datelor din memorie catre procesor

  - store reg $\rightarrow$ mem
  - load reg $\leftarrow$ mem

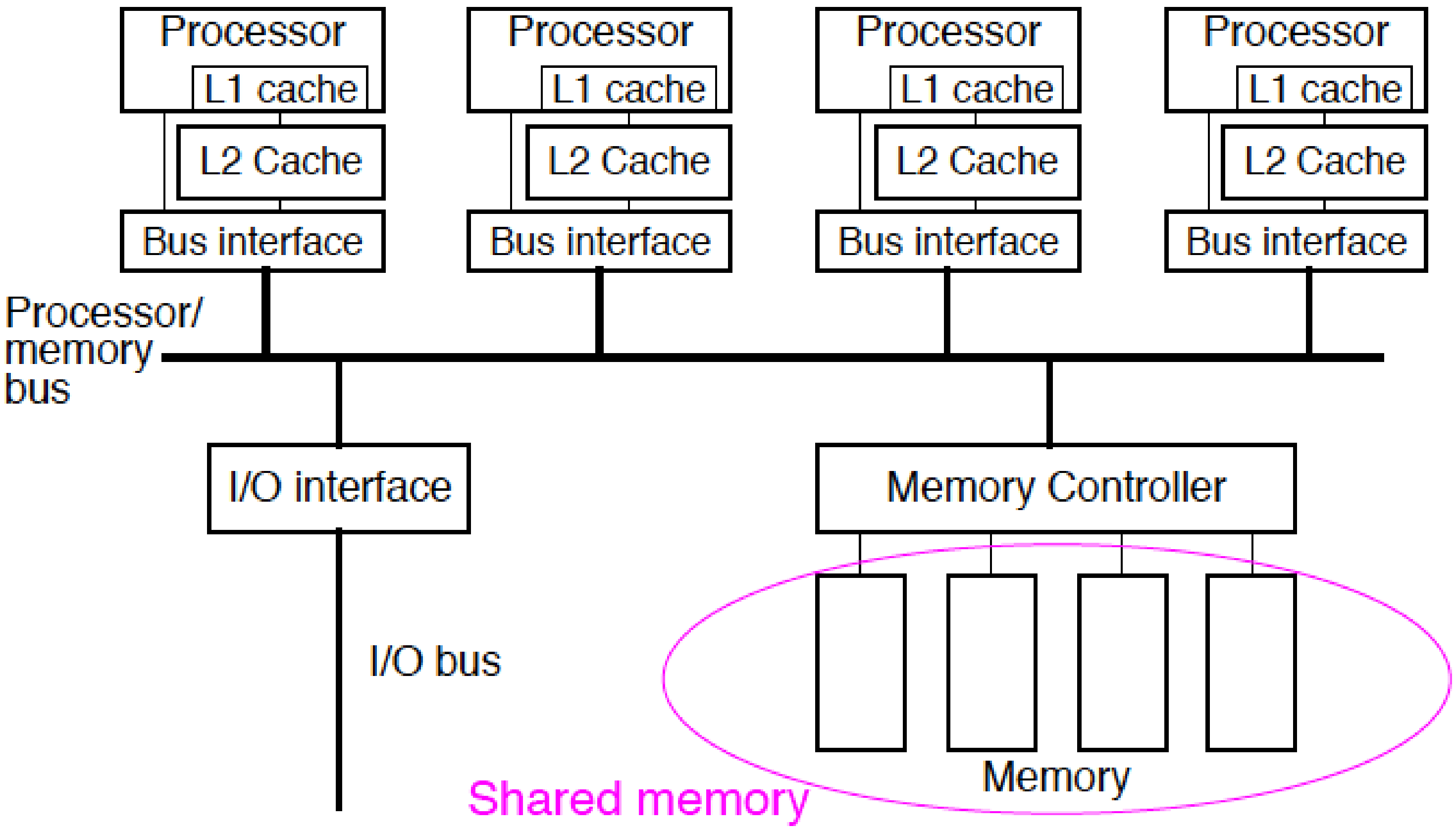
Caching in sistemele cu memorie partajata

- Folosirea memoriilor cache intr-un system de tip SMP introduce probleme legate de ***cache coherency:***
  - Cum se garanteaza faptul ca atunci cand o data este modificata, aceasta modificare este reflectata in celelalte memorii cache si in main memory?

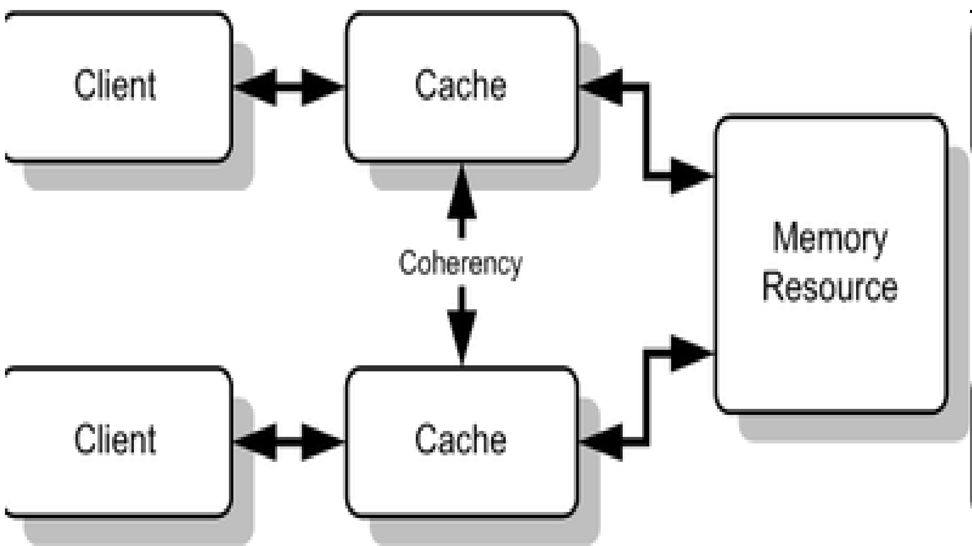


- *coherency* diminueaza scalabilitatea
  - shared memory systems=> maximum 60 CPUs (2016).

Niveluri de caching



Cache coherence



Cache Coherency <-> SMP

- Memoriile cache sunt foarte importante in SMP pentru asigurarea performantei
  - Reduce timpul mediu de acces la date
  - Reduce cerinta pentru largime de banda- *bandwidth*- plasate pe interconexiuni partajate
- Probleme coresp. *processor caches*
  - Copiile unei variabile pot fi prezente in cache-uri multiple;
  - o scriere de catre un procesor poate sa nu fie vizibila altor procesoare
    - acestea vor avea valori vechi in propriile cache-uri

  $\Rightarrow$ *Cache coherence* problem
- Solutii:
  - organizare ierarhica a memoriei;
  - Detectare si actiuni de actualizare.

Motivatii pentru asigurarea consistentei memoriei

- Coerenta implica faptul ca scrierile la o locatie devin vizibile tuturor procesoarelor in aceeasi ordine .
- cum se stabileste ordinea dintre o citire si o scriere?
    - Sincronizare (*event based*)
  - Implementarea unui protocol hardware pentru *cache coherency.*
  - Protocolul se poate baza pe un model de consistenta a memoriei.

simplist

P_1	P_2
/* Assume initial value of A and flag is 0 */	
`A = 1;`	`while (flag == 0);` /* spin idly */
`flag = 1;`	`print A;`

Asigurarea consistentei memoriei

- Specificare de constrangeri legate de ordinea in care operatiile cu memoria pot sa se execute.
- Implicatii exista atat pentru programator cat si pentru proiectantul de sistem:
  - programatorul le foloseste pentru a asigura corectitudinea ;
  - proiectantul de sistem le poate folosi pentru a constrange gradul de reordonare a instructiunilor al compilatorului sau al hardware-ului.
- Contract intre programator si sistem.

(Consistenta secventiala) Sequential Consistency

- Ordine totala prin intreteserea acceselor de la diferite procesoare
  - *program order*
  - Operatiile cu memoria ale tuturor procesoarelor par sa inceapa, sa se execute si sa se termine 'atomic' ca si cum ar fi doar o singura memorie (no cache).

*"A multiprocessor is **sequentially consistent** if the **result** of **<u>any execution</u>** is the same as if the operations of all the processors were executed in <u>some sequential order</u>, and the operations of each individual processor appear in this sequence in the order specified by its program."*
[L. Lamport, 1979]

Shared Memory

Avantaje:

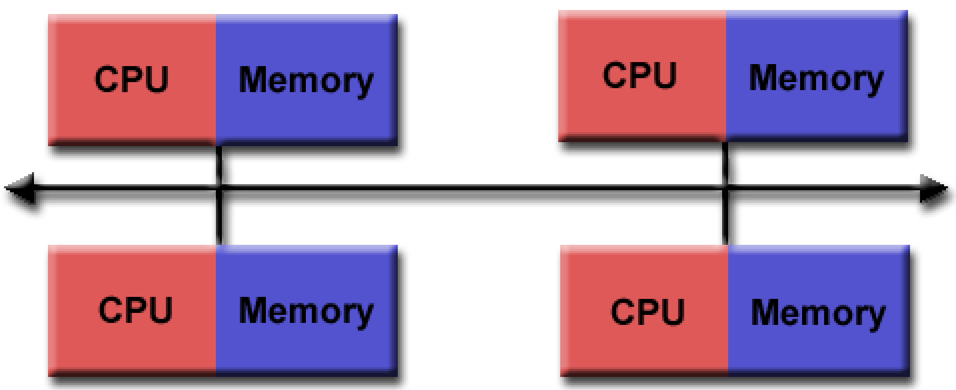
- *Global address space*
- *Partajare date rapida si uniforma*

Dezavantaje:

- Lipsa scalabilitatii
- Sincronizare in sarcina programatorului
- Costuri mari

Arhitecturi cu Memorie Distribuita/*Distributed Memory*

- Retea de interconectivitate /***communication network***
- Procesoare cu memorie locala ***local memory***.



Arhitecturi cu memorie distribuita

Avantaje:

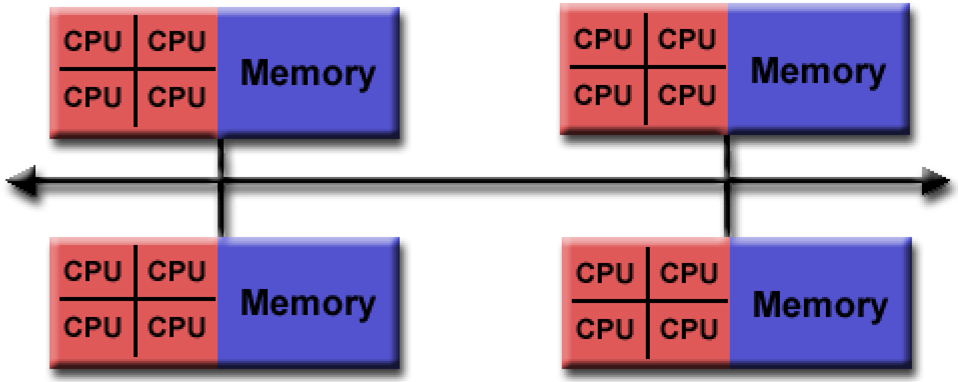
- Memorie scalabila – odata cu cresterea nr de procesoare
- Cost redus – retele

Dezavantaje:

- Responsibilitatea programatorului sa rezolve comunicatiile.
- Dificil de a mapa structuri de date mari pe mem. distribuita.
- Acces Ne-uniform la memorie

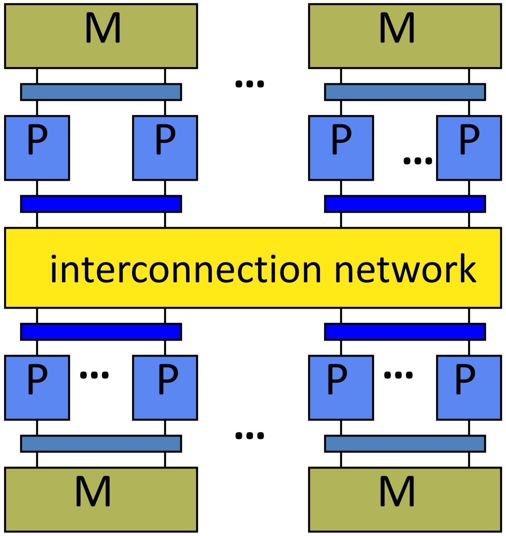
Hybrid Distributed-Shared Memory

- Retea de SMP-uri



SMP Cluster

- Clustering
  - Noduri integrate
- Motivare
  - Partajare resurse
  - Se reduc costurile de retea
  - Se reduc cerintele pt largimea de banda (*bandwidth*)
  - Se reduce latenta globala
  - Creste performanta per node
  - Scalabil



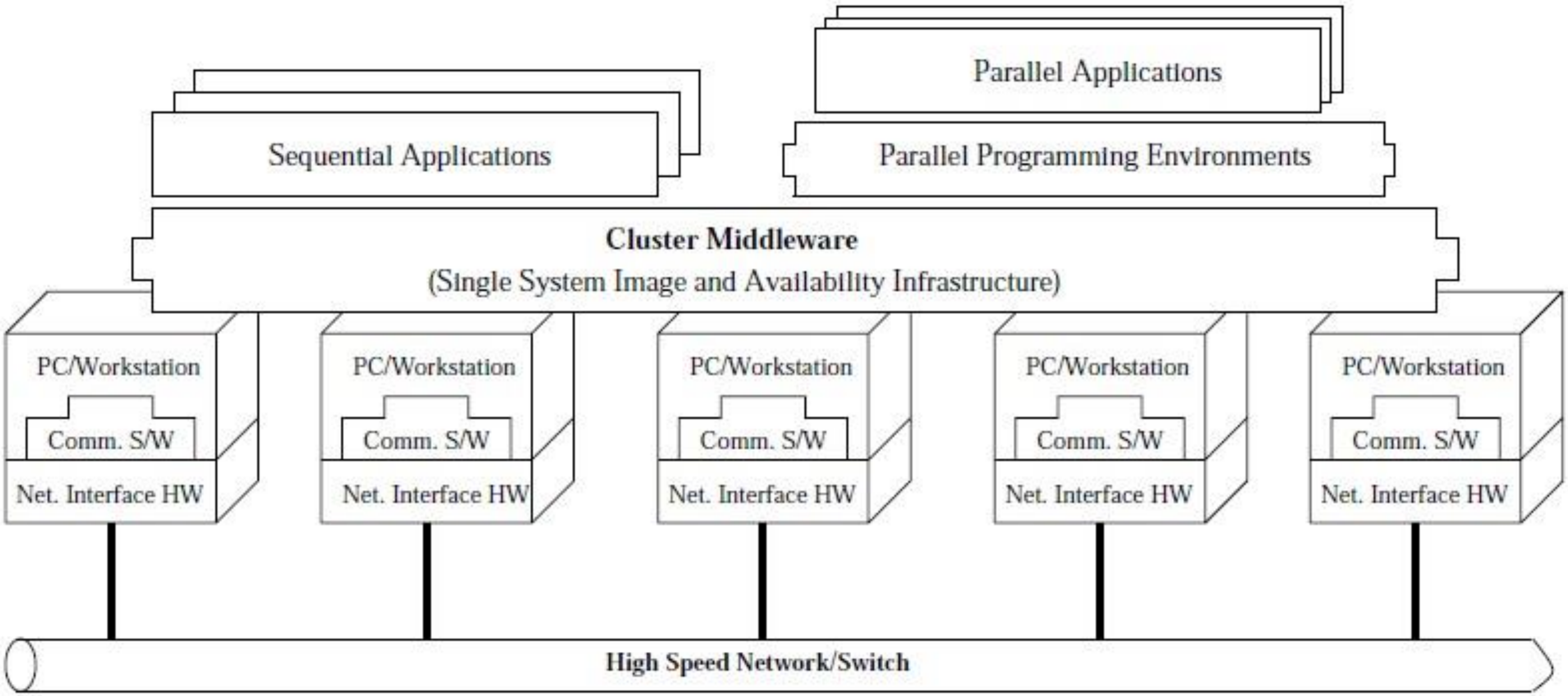
MPP(Massively Parallel Processor)

- Fiecare nod este un sistem independent care are local:
  - Memorie fizica
  - Spatiu de adresare
  - Disc local si conexiuni la retea
  - Sistem de operare

- *MPP (massively parallel processing) is the coordinated processing of a program by multiple processors that work on different parts of the program, with **each processor using its own operating system and memory.** Typically, MPP processors communicate using some messaging interface. In some implementations, up to 200 or more processors can work on the same application. An "interconnect" arrangement of data paths allows messages to be sent between processors. Typically, the setup for MPP is more complicated, requiring thought about how to partition a common database among and how to assign work among the processors. An MPP system is also known as a "loosely coupled" or "shared nothing" system.*
- *An MPP system is considered better than a symmetrically parallel system (SMP) for applications that allow a number of databases to be searched in parallel. These include decision support system and data warehouse applications.*

COW

- Cluster of Workstations



Scalabilitatea sistemelor de calcul

- Cat de mult se poate mari sistemul?
  - unitati de procesare,
  - unitati de memorie

- Cate procesoare se pot adauga fara a se diminua caracteristicile generale ale acestuia (viteza de comunicare, viteza de accesare memorie, etc.)

- Masuri de eficienta (*performance metrics*)


SMP grows by buying
a bigger system.
SMP


MPP grows by adding
to the existing system.
PDW (MPP)

Performanta

- FLOP= floating point operation
- FLOPS: metrica de performanta => floating point operations per second
  - pentru un calculator
    - FLOPS= cores*cycles_per_sec*FLOPs_per_cycle

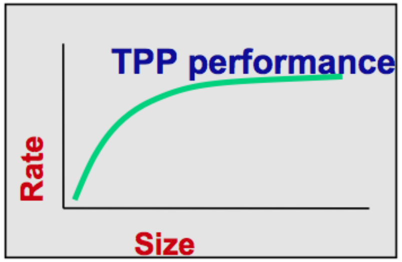
- 32-bit (*FP32*) single precision and 64-bit (*FP64*) double precision operations

- Problema: daca un procesor este evaluat la nivel k MFLOPS si sunt p procesoare, este performanta totala de ordin k*p MFLOPS?
- Problema: daca un calcul necesita 100 sec. pe un procesor se va putea face in 10 sec. pe 10 procesoare?
- Cauze care pot afecta performanta
  - Fiecare proc. –unitate independenta
  - Interactiunea lor poate fi complexa
  - *Overhead ...*

- *Need to understand performance space*

Top 500 Benchmarking
https://www.top500.org/project/linpack/

- Cele mai puternice 500 calculatoare din lume
- High-performance computing (HPC)
  - Rmax : *maximal performance Linpack benchmark*
    - Sistem dens liniar de ecuatii (Ax = b)



- Informatii date
  - Rpeak : *theoretical peak performance*
    - product of the total number of CPU cores, the core clock frequency the number of FLOPs one core makes per clock tick, parallel instruction issuing capacity, vector register sizes, etc. which are design characteristics of the core. This performance can never be reached in practice.
  - Nmax : dimensiunea problemei necesara pt a se atinge Rmax
  - N1/2 : dimensiunea problemei necesara pt a se atinge 1/2 of Rmax
  - Producator si tipul calculatorului
  - Detalii legate de instalare (location, an,…)
- Actualizare de 2 ori pe an

UBB CLUSTER – IBM Intelligent Cluster
http://hpc.cs.ubbcluj.ro/

- Hybrid architecture
  - HPC system +
  - private cloud

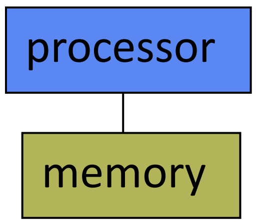
SUMARIZARE

Vedere actuala asupra tipurilor de arhitecturilor paralele

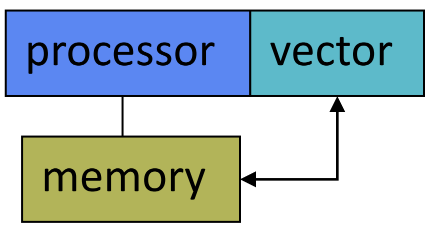
Parallel Architecture Types

imagini preluate de la course pres. Introduction to Parallel Computing CIS 410/510, Univ. of Oregon

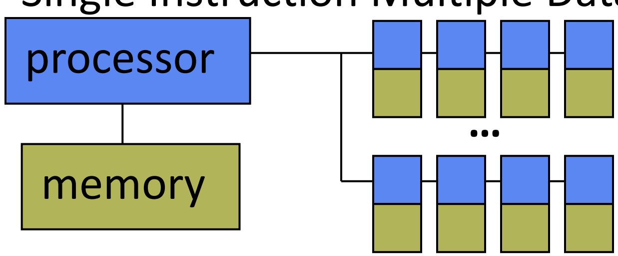
- Uniprocessor
  - Scalar processor



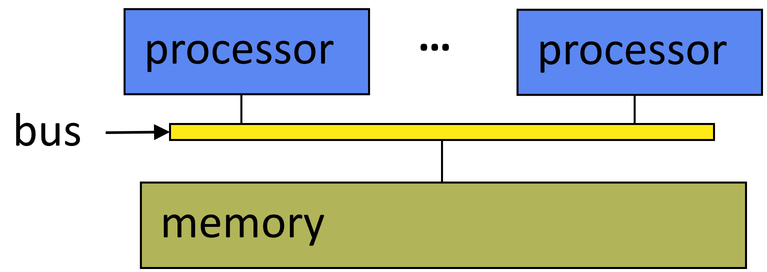
  - Vector processor



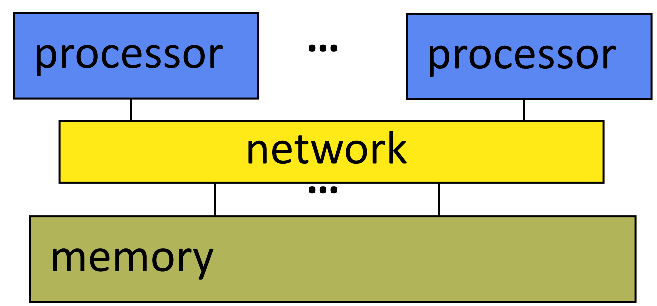
  - Single Instruction Multiple Data



- Shared Memory Multiprocessor (SMP)
  - Shared memory address space
  - Bus-based memory system

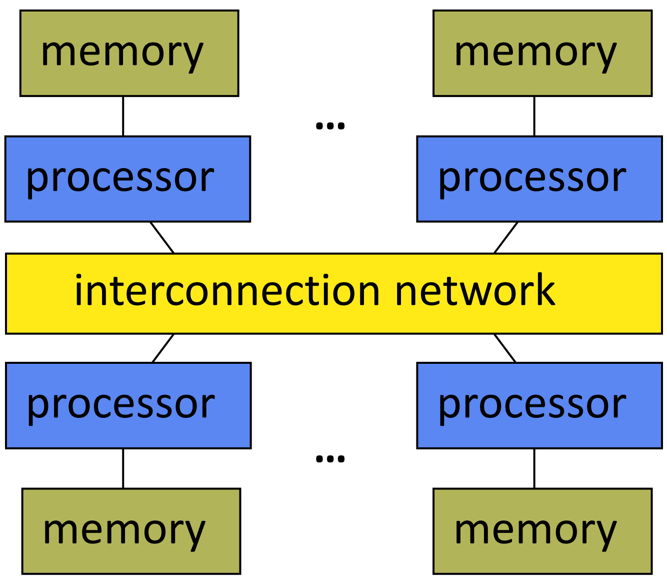


  - Interconnection network



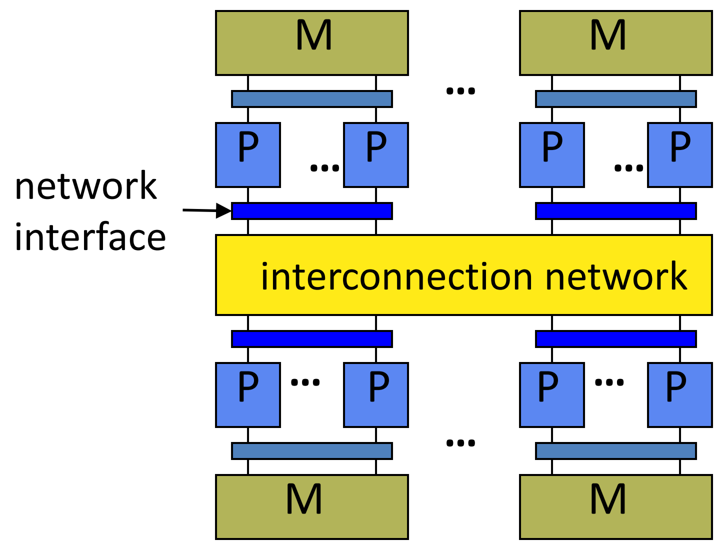
Parallel Architecture Types (2)

- Distributed Memory Multiprocessor
  - Message passing between nodes



  - Massively Parallel Processor (MPP)
    - many, many processors
    - fast interconnection

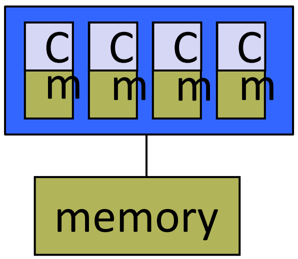
- Cluster of SMPs
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes



  - Can also be regarded as MPP if processor number is large and the communication is fast

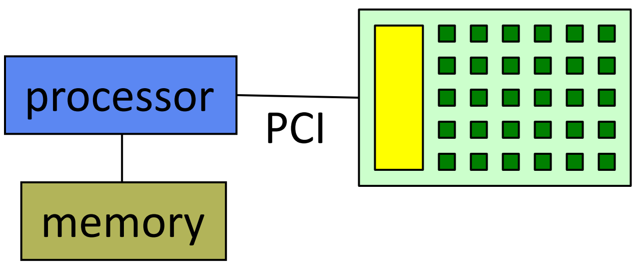
Parallel Architecture Types (3)

- Multicore
  - Multicore processor

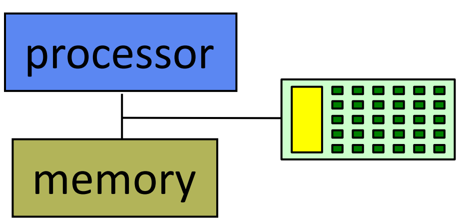


cores can be hardware multithreaded (hyperthread)

  - GPU accelerator



  - "Fused" processor accelerator



- Multicore SMP+GPU Cluster
  - Shared memory addressing within SMP node
  - Message passing between SMP nodes
  - GPU accelerators attached