Automi Cellulari vs Equazioni Differenziali

Everything should be made as simple as possible, but not simpler !
(Albert Einstein)

Cellular Automata is an “alternative”, rather than an “approximation”, to Differential Equations
(Tommaso Toffoli)

Introduzione alle Scienze computazionali

- Consentono di predire fenomeni fisici dettagliati a partire dalla teoria di base
- Utilizzano la matematica e l'informatica per estendere le teorie fondamentali delle scienze fisiche, biologiche e sociali alla descrizione di sistemi fisici complessi.

L'uso dei calcolatori per studiare sistemi fisici permette di gestire fenomeni

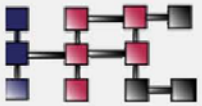
- **molto grandi** (meteo-climatologia, cosmologia, data mining, oil reservoir)
- **molto piccoli** (drug design, silicon chip design, biologia strutturale)
- **molto complessi** (fisica delle particelle elementari, dinamica dei fluidi)
- **molto pericolosi o costosi** (simulazione di guasti, test nucleari, crash analysis)

La simulazione al calcolatore attualmente è largamente accettata come la terza modalità per fare ricerca: Teoria, Esperimenti, Simulazione

HPCC

GRAND CHALLENGES

Le grandi sfide computazionali



Gli scienziati pensano di utilizzare i computer per studiare
i Grandi Problemi Fondamentali della scienza e dell'ingegneria

- Modellizzazione globale dell'ambiente
- Modellizzazione biologica
- chimica computazionale
- quanto Cromo Dinamica (QCD)
- modellazione semiconduttori
- superconduttività
- simulazione globale del volo
- combustione e turbolenza dei fluidi
- visione artificiale e cognizione
- modelli globali di sistemi socio-economici

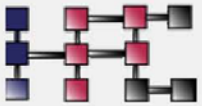
Questi grandi problemi manifestano una complessità crescente in termini di potenza di calcolo, quantità di memoria e strumenti di post-elaborazione dei risultati (visualizzazione scientifica)

Units of High Performance Computing

1 Mflop/s	1 Megaflop/s	10^6 Flop/sec
1 Gflop/s	1 Gigaflop/s	10^9 Flop/sec
1 Tflop/s	1 Teraflop/s	10^{12} Flop/sec
1 Pflop/s	1 Petaflop/s	10^{15} Flop/sec
1 MB	1 Megabyte	10^6 Bytes
1 GB	1 Gigabyte	10^9 Bytes
1 TB	1 Terabyte	10^{12} Bytes
1 PB	1 Petabyte	10^{15} Bytes

HPCC

Un esempio: previsioni climatiche su scala globale



- Serve a prevedere eventi notevoli, es. El Niño e la diffusione e l'immissione di sostanze inquinanti
- Problema: risolvere le equazioni di Navier-Stokes
- Richieste di potere computazionale:
  - griglia di 1 km di lato: circa 5×10^{10} celle
  - richiesta di memoria circa 2TByte
  - per una previsione a 24 ore occorrono 1.5×10^{17} operazioni
- Un calcolatore con potenza di 1 TFLOP/s impiegherà 1.5×10^{5} sec cioè quasi 2 giorni

Previsioni del tempo

- L'intera atmosfera è modellata dividendola in regioni o *celle* tridimensionali
- Il calcolo su ogni cella è ripetuto diverse volte per modellare il passaggio del tempo
- Esempio:
  - l'atmosfera è suddivisa in celle di dimensioni *1 miglio × 1 miglio × 1 miglio* per un'altezza di *10 miglia* (per un'altezza di *10 celle*)
  - per un totale di circa ***5 × 10^8 celle***
- Supponiamo che il calcolo associato a ciascuna cella richiede 200 FP ops, in un passo sono necessari ***10^{11} FP ops***
- Per una previsione meteo su 10 giorni, usando intervalli di tempo di simulazione di 10 minuti, sono necessari 6 passi di simulazione all'ora, per un totale di *1440 passi* (circa *1,4 10^{14}* FP ops)
  - Un computer che operasse a **100 Mflops** (10^8 FP ops/sec) impiegherebbe più di 10^6 secs (circa 16 giorni !!!!).
  - Per eseguire lo stesso calcolo in 10 minuti ci vorrebbe un computer in grado di eseguire **0,2 Tflops** ($0,2 \times 10^{12}$ FP ops/sec)
- Cosa succede se aumentiamo i passi o rimpiccioliamo le celle?

Sistemi Complessi

- Le caratteristiche associate con Sistemi Complessi riguardano la presenza di un numero elevato di elementi interagenti, interazione non lineare e presenza di comportamenti emergenti, con nessun analogo microscopico corrispondente. Infine, capacità auto-organizzative

- Gli Automi Cellulari, insieme alle Reti Neurali ed Algoritmi Genetici, rappresentano strumenti validi per la descrizione di fenomeni complessi

Introduzione al Calcolo Parallelo

- E’ proprio necessario? ES:Panda o Ferrari?
- Motivazioni: Risparmiare tempo e risolvere problemi maggiori
- MFlops, TFlops, PFlops?
- Esempio: Previsioni del Tempo!
- Come fare?
  1. Connettere piu’ elementi di calcolo (CPUs)
  2. Software appropriato
  3. Algoritmi e Strutture Dati
  4. Suddivisione degli Algoritmi e Strutture Dati
  5. Identificare le comunicazioni tra i sottoproblemi (processi)
  6. Assegnare i sottoproblemi ai processori e memoria

Motivazioni

- **Limite di prestazioni delle architetture sequenziali** (velocità di propagazioni dei segnali, limiti alla miniaturizzazione dei componenti elettronici), limiti economici, memory bottleneck, etc

- **Miglior rapporto costo/prestazioni** (Beowulf: cluster di workstation)

- **Grande varietà** di architetture proposte e realizzate

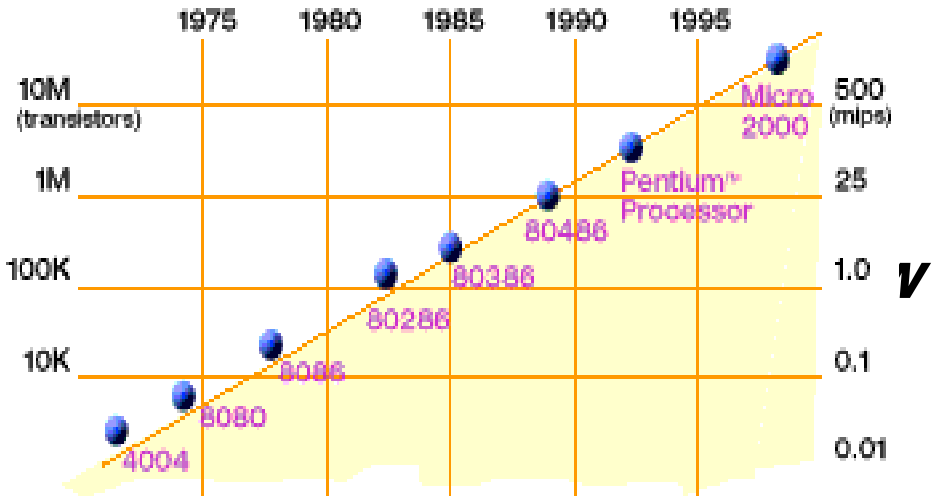
MA...

- Non sempre si riesce a sfruttarne le potenzialità adeguatamente! Infatti la VERA difficoltà non è realizzare architetture parallele, ma **SVILUPPARE APPLICAZIONI PARALLELE!!!**

Prestazioni

- Esistono due modi per migliorare le prestazioni
  - Un problema più grande nello stesso tempo (SCALE-UP)
  - Lo stesso problema in minor tempo (SPEED-UP)
- 1024 processori x 200 Mflops/processori = (teoricamente) 200 GFlops
- *Il **calcolo parallelo** è l'unico strumento in grado di affrontare le grandi simulazioni del mondo fisico (Grand Challenge Problems) quali la meteorologia, genoma, etc*

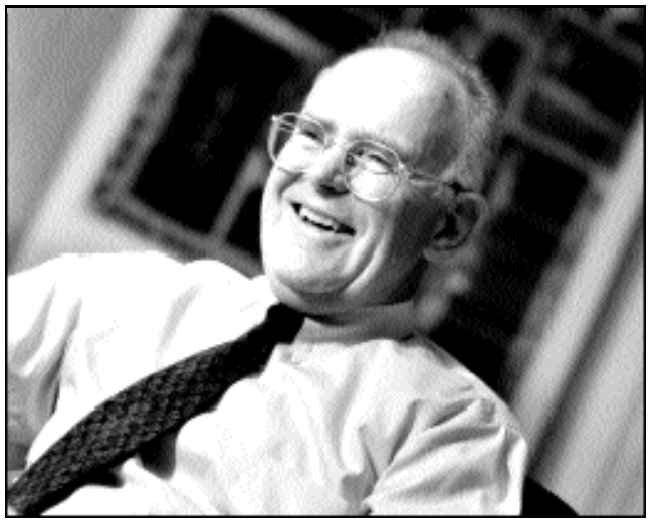
Technology Trends: Capacità del Microprocessore



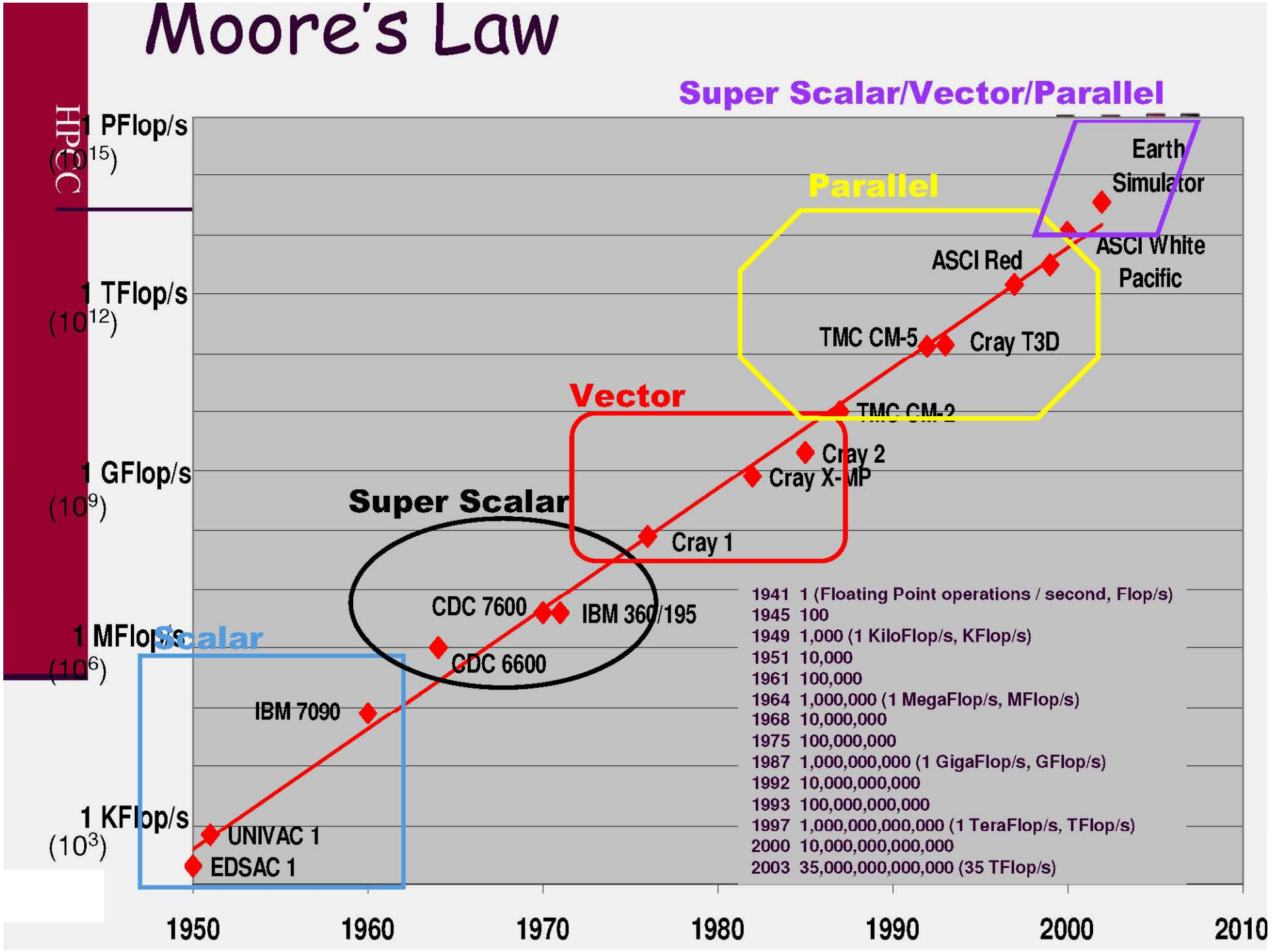
2X transistors/Chip ogni 1.5 anni

"***Moore's Law***"

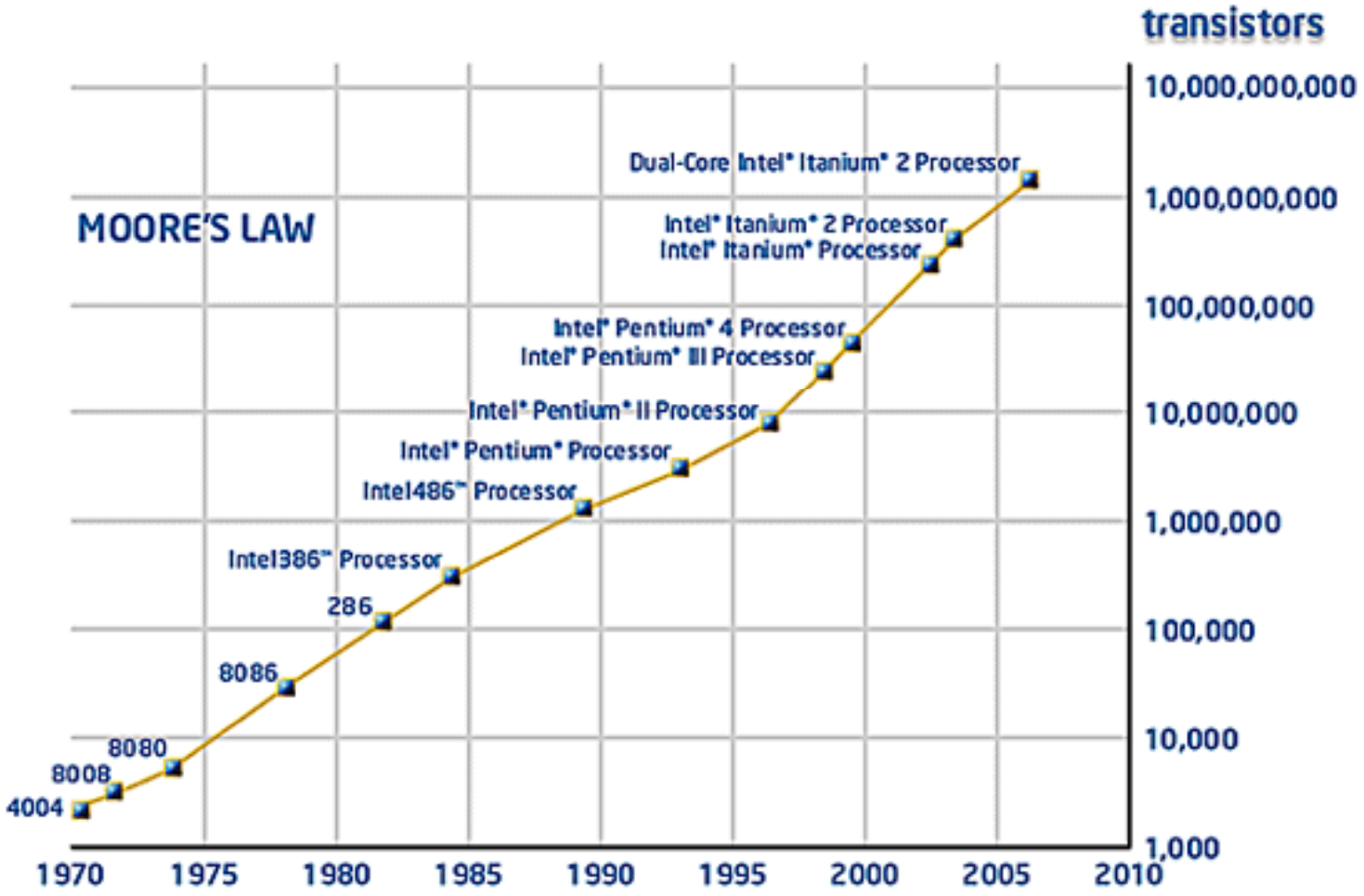
I microprocessori sono diventati piu' piccoli, piu' densi, e piu' potenti!



Gordon Moore (cofondatore della Intel) predisse nel 1965 che la densità dei transitor nei chip sarebbe raddoppiato ogni 18 mesi.


Moore's Law
Super Scalar/Vector/Parallel
HPEC
1 PFlop/s
(10^15)
1 TFlop/s
(10^12)
1 GFlop/s
(10^9)
1 MFlop/s
(10^6)
1 KFlop/s
(10^3)
Parallel
Vector
Super Scalar
Scalar
Earth Simulator
ASCI White Pacific
ASCI Red
TMC CM-5
Cray T3D
TMC CM-2
Cray 2
Cray X-MP
Cray 1
CDC 7600
IBM 360/195
CDC 6600
IBM 7090
UNIVAC 1
EDSAC 1
1941 1 (Floating Point operations / second, Flop/s)
1945 100
1949 1,000 (1 KiloFlop/s, KFlop/s)
1951 10,000
1961 100,000
1964 1,000,000 (1 MegaFlop/s, MFlop/s)
1968 10,000,000
1975 100,000,000
1987 1,000,000,000 (1 GigaFlop/s, GFlop/s)
1992 10,000,000,000
1993 100,000,000,000
1997 1,000,000,000,000 (1 TeraFlop/s, TFlop/s)
2000 10,000,000,000,000
2003 35,000,000,000,000 (35 TFlop/s)
1950
1960
1970
1980
1990
2000
2010

Transistors dei Microprocessori

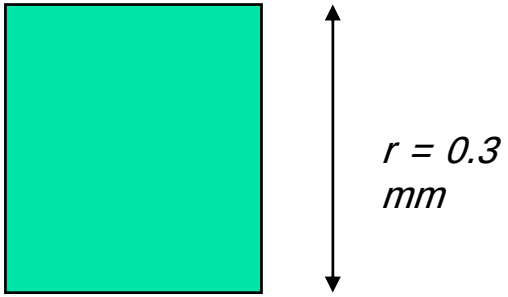


Eniac and My Laptop

	Eniac	My Laptop
Year	1945	2002
Devices	18,000	6,000,000,000
Weight (kg)	27,200	0.9
Size (m^3)	68	0.0028
Power (watts)	20,000	60
Cost (1999 dollars)	4,630,000	1,000
Memory (bytes)	~200	1,073,741,824
Performance (FP/sec)	800	5,000,000,000

Quanto veloce può essere un calcolatore sequenziale?

1 Tflop/s, 1 Tbyte macchina sequenziale



- Consideriamo una macchina (ipotetica!) a 1 Tflop/s (10^{12} flop/s):
  - I dati devono percorrere una certa distanza, r, per andare dalla memoria alla CPU.
  - Se possiamo prelevare 1 elemento-data per ciclo, significa 10^{12} volte al secondo alla velocità della luce ($c = 3 \times 10^8$ m/s), si ha che $r < c/10^{12} = 0.3$ mm.
- Mettiamo ora 1 Tbyte di memoria in un'area 0.3 mm x 0.3 mm:
  - Otteniamo che ogni parola occupa 3 Angstroms quadrati, ovvero le dimensioni di un ATOMO!

Terminologia ...

Task Parallelo

Un Task che può essere eseguito in modo “sicuro” su più processori (risultati corretti!)

Esecuzione Parallela

Esecuzione di un programma tramite più task, con ognoi task capace di eseguire la stessa o differente istruzione allo stesso istante

Memoria Condivisa

Architettura di computer dove ogni processore ha accesso diretto (e.g. bus) ad una memoria comune fisica. Dal punto di vista della programmazione, esso descrive un modello dove task paralleli hanno lo stesso “schema” di memoria e possono indirizzare ed accedere le stesse locazioni logiche, a prescindere da dove si trova realmente la memoria fisica

Memoria Distribuita

In hardware, si riferisce ad accessi a memoria fisica no comune basati su network. Come modello di programmazione, i task possono solamente “vedere” memoria locale e devono usare qualche tipo di **comunicazione** per accedere alla memoria di altre macchine mentre altri task sono in esecuzione

Terminologia ...

Comunicazioni

In generale, task paralleli necessitano di scambiare dati. Esistono diversi modi di ottenere ciò: tramite una **memoria condivisa** (o bus) o via **rete**

Speedup

$$S = \frac{\text{tempo sequenziale}}{\text{tempo parallelo}}$$

Rappresenta uno degli indicatori più semplici e usati per misurare la performance di un programma parallelo

Scalabilità

- Si riferisce alla abilità di un sistema parallelo (hardware e/o software) di mostrare un incremento proporzionale nello speedup se si aumenta il numero di processori. Fattori che contribuiscono ad una buona scalabilità sono:
  - Hardware : bandwith di memoria/cpu e comunicazioni di rete
  - algoritmo impiegato
  - Overhead Parallelo (prezzo da pagare! Es: comunicazioni)
  - Caratteristiche del problema specifico e codifica

Opzioni di Progetto

Processori

- Numero, tipo, capacità di elaborazione
- Da semplici ALU a CPU complete
- Da poche unità a molte migliaia

Memorie

- Private o condivise
- Schemi complessi di *caching*

Interconnessione

- Topologia di interconnessione
- Numero di collegamenti
- Distanza tra gli elementi
- Scalabilità dell'architettura

Modelli di Comunicazione

Due approcci fondamentali:

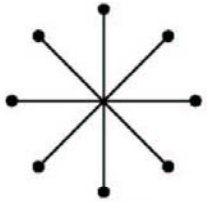
Multiprocessori

- Tutte le CPU condividono una memoria fisica comune
- Spazio comune di memoria virtuale
- I processi possono comunicare tramite la memoria
- *Accoppiamento stretto*

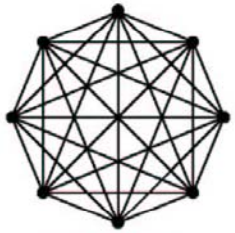
Multicomputers

- Ogni CPU ha la sua memoria privata
- Uno spazio di memoria fisica per ciascuna CPU
- I processi comunicano tramite scambio di messaggi
- *Accoppiamento lasco*

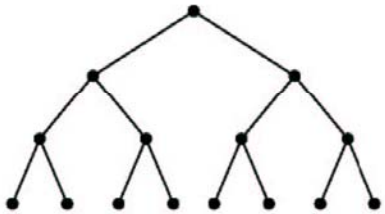
Topologie di Interconnessione



STELLA

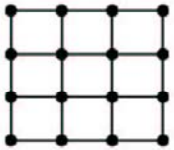


COMPLETA

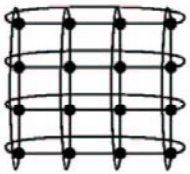


ALBERO

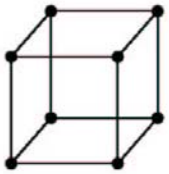
ANELLO



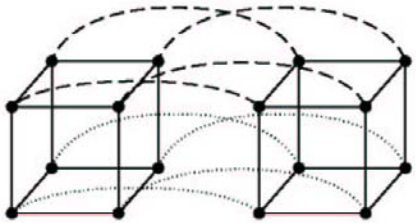
GRIGLIA



DOPPIO TORO



CUBO



IPERCUBO

Topologie di Interconnessione (2)

Stella

- Tutte le comunicazioni attraversano un nodo
- Distanza costante tra tutti i nodi
- Non scalabile: fan-out lineare

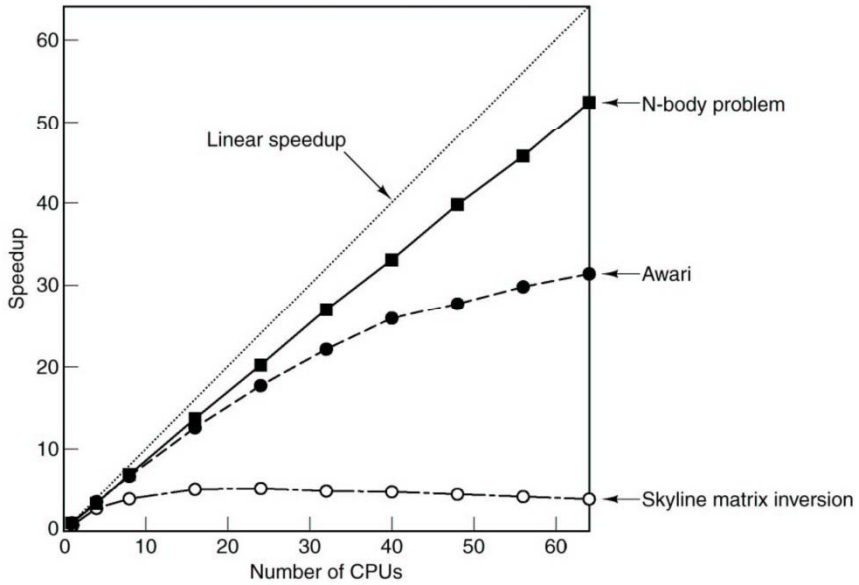
Completa

- Il numero di interconnessioni cresce con il quadrato dei nodi n^2
- Distanza costante tra tutti i nodi
- Non scalabile: fan-out lineare

Albero

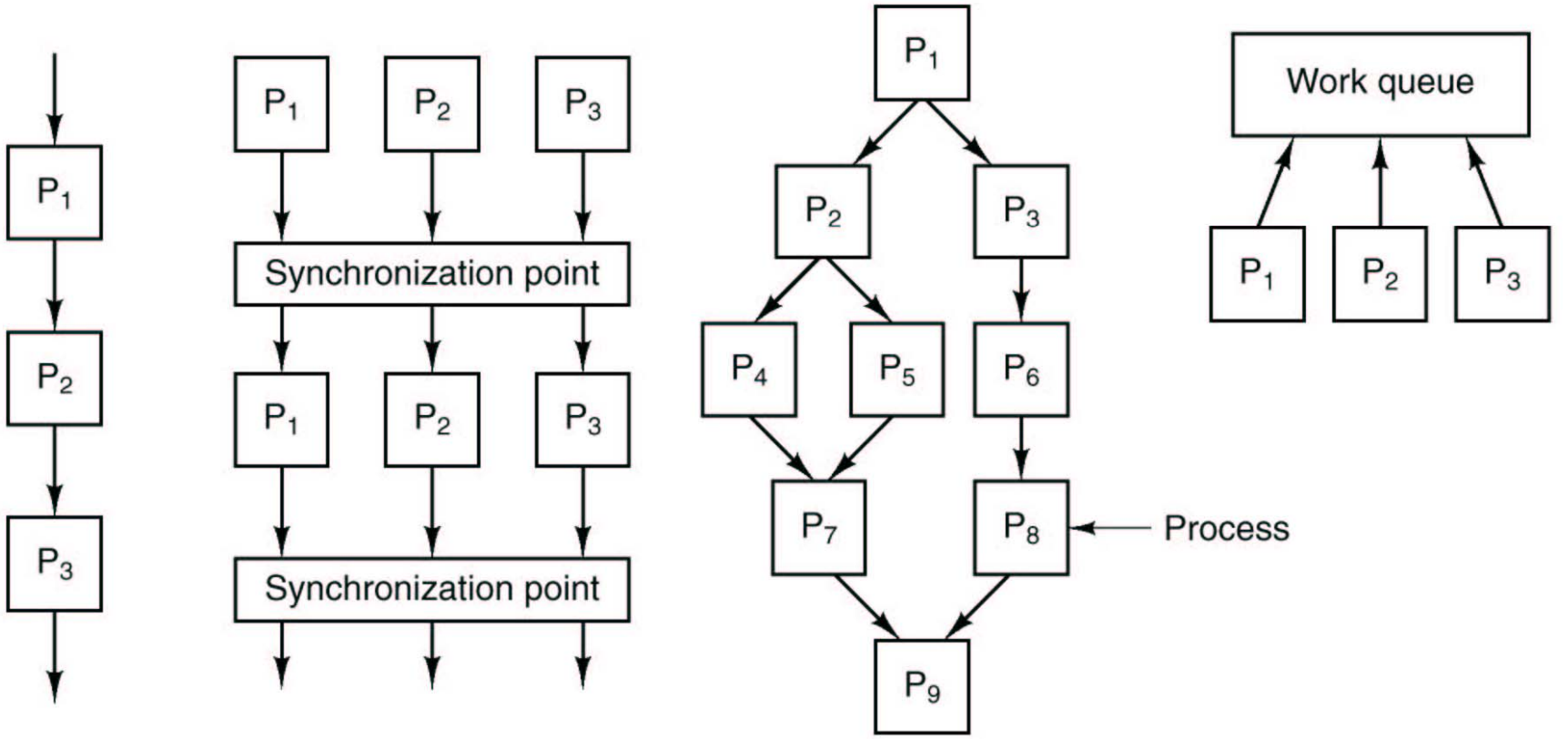
- Profondità logaritmica: $\log_r n$
- Distanza logaritmica tra i nodi
- Fan-out r costante
- Non scalabile: rami alti congestionati

Metriche di Prestazione



- *Speedup*: fattore di velocità che si guadagna nell'esecuzione rispetto ad un uniprocessore
- Caso ideale: *speedup lineare* con il numero dei processori
- Purtroppo è raramente raggiunto
- Dipende sia dalle applicazioni che dai vincoli dell'architettura

Paradigmi Computazionali



PIPELINE PHASED COMPUTATION DIVIDE ET IMPERA REPLICATED WORKER

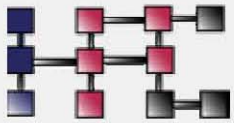
Tassonomia di Flynn

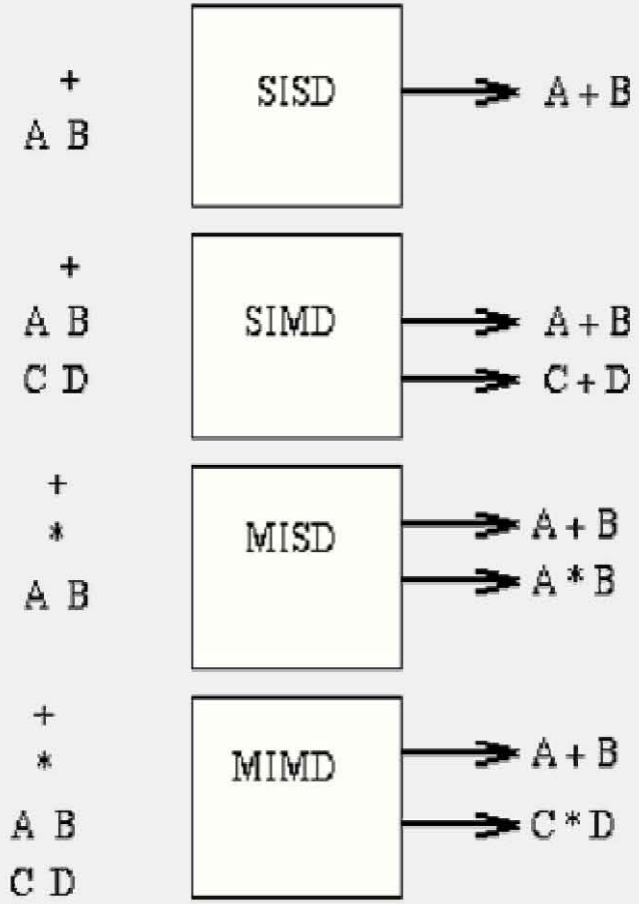
Instruction streams	Data streams	Name	Examples
1	1	SISD	Classical Von Neumann machine
1	Multiple	SIMD	Vector supercomputer, array processor
Multiple	1	MISD	Arguably none
Multiple	Multiple	MIMD	Multiprocessor, multicomputer

- Introdotta da Flynn, basata sui concetti di *instruction stream* e *data stream,* e considera tutte le possibili combinazioni:

- **SISD**: Single Instruction Single Data
- **SIMD**: Single Instruction Multiple Data (la stessa operazione effettuata su dati diversi)
- **MISD**: Multiple Instruction Single Data (finora nessuna implementazione)
- **MIMD**: Multiple Instruction Multiple Data (operazioni distinte su dati distinti)
- Estensione della tassonomia per considerare diversi tipi di architetture MIMD

Potenzialità dei 4 modelli di computer





Single Instruction, Single Data (SISD)

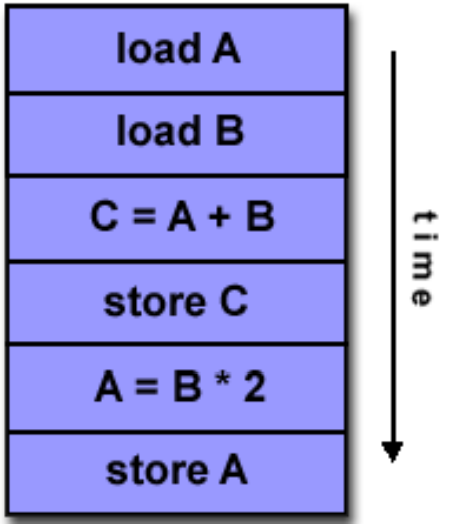
- *Un calcolatore (non-parallelo) sequenziale*

- ***Single instruction**: solo un flusso di istruzioni viene eseguito durante un ciclo macchina*

- ***Single data**: solo un flusso di dati viene usata come input durante un ciclo macchina*

- ***Esecuzione deterministica***

- ***Esempi**: la maggior parte di PCs, workstations a singola CPU e mainframes*

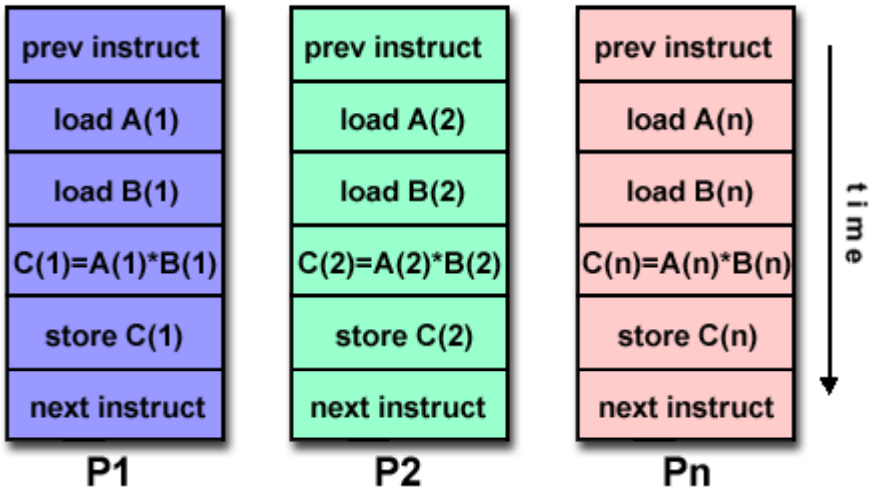


Single Instruction, Multiple Data (SIMD)

- ***Single instruction***: *Tutti le unità di calcolo eseguono la stessa istruzione ad ogni ciclo clock*

- ***Multiple data***: *Ogni unità di elaborazione può operare su un elemento data diverso*

- *Tipicamente, questa macchina ha un dispatcher di istruzioni, una rete interna ad elevatissima bandwith, e un grande numero di piccole unità per istruzioni*



- *Utilizzati principalmente per problemi "specializzati" caratterizzati da un elevato grado di regolarità, come l'image processing.*

- ***Esecuzione sincrona (lockstep) e deterministica***

- ***Esempi***:
  - o*Processor Arrays: Connection Machine CM-2, Maspar MP-1, MP-2*
  - o*Vector Pipelines: IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820*
  - o*SSE (Streaming SIMD Extensions) for Pentium Intel*

Multiple Instruction, Single Data (MISD)

- *In realtà pochi esempi pratici esistono*

- *Alcuni esempi "potrebbero" essere:*
  - *Filtri multipli di frequenza che operano su un singolo flusso di segnale*

  - *Algoritmi multipli di crittografia che cercano di crackare un messaggio singolo…*

Multiple Instruction, Multiple Data (MIMD)

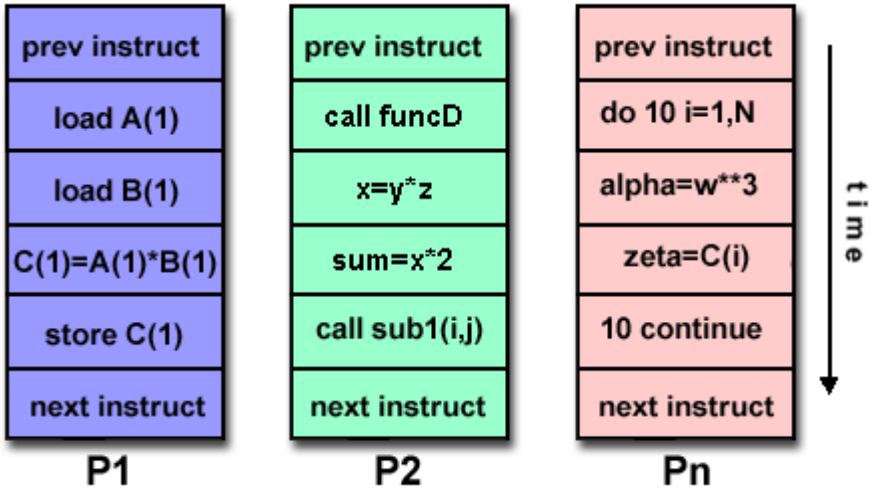
- *Il più comune tipo di computer parallelo*

- ***Multiple Instruction**:*
Ogni processore può eseguire uno stream differente di istruzioni

- ***Multiple Data**:*
Ogni processore può lavorare con un diverso stream di dati

- *L'esecuzione può essere **sincrono o asincrono**, **deterministico o non- deterministico***

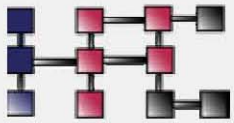
- ***Esempi**: supercomputers attuali, "grid" di computer paralleli basati su network e computers multi-processore SMP, alcuni tipi di PC (e.g. Dual Core)*



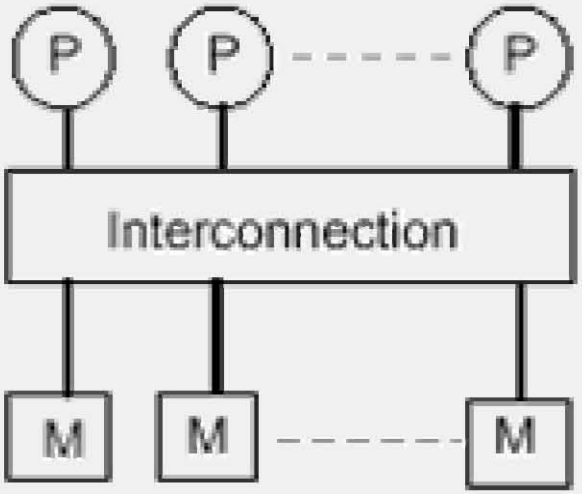
Architetture MIMD

- **UMA (Uniform Memory Access)**
  - Tutte le CPU hanno lo stesso tempo di accesso a tutta memoria
  - Allocazione dei dati irrilevante
- **NUMA (NonUniform Memory Access)**
  - Per ciascuna CPU il tempo di accesso varia a seconda del modulo acceduto
  - Allocazione dei dati critica
- **COMA (Cache Only Memory Access)**
  - L'accesso avviene solo tramite cache
- **MPP (Massive Parallel Processors)**
  - Sistemi autonomi (CPU-Memoria) connessi con reti proprietarie veloci
- **COW (Cluster Of Workstations)**
  - Insiemi di workstation connessi con tecnologia *off-the-shelf*

Multiprocessori Shared Memory



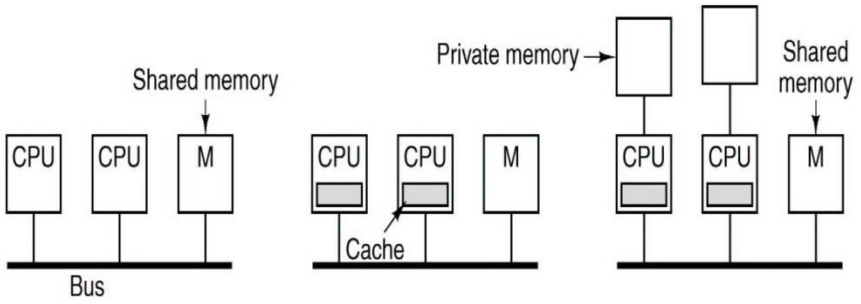
- Le operazioni Load/Store accedono direttamente a tutte le locazioni di memoria del sistema
  - meccanismo semplice per comunicare e condividere
    - trasparente rispetto alla locazione
    - stesse operazioni supportate dagli uniprocessori
- Memoria centralizzata o distribuita vicino ai processori
- Costo dei Memory Access *Uniformi* (UMA) o *Non Uniformi*: (NUMA)



UMA

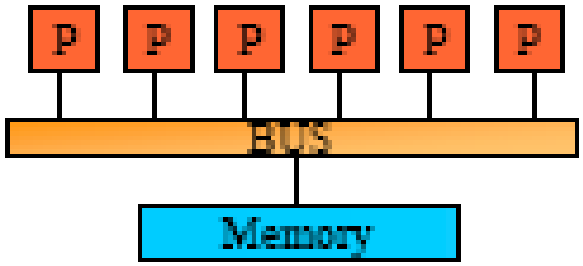
NUMA

Architetture UMA a Bus



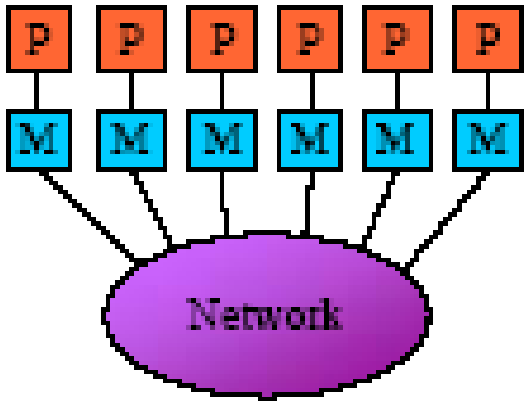
- Il Bus può divenire collo di bottiglia, e comunque limita la scalabilità
- Le memorie private alleviano il problema
- Anche le cache diminuiscono il traffico sul Bus
- *Problema della coerenza di cache*
- Se lo stesso blocco è presente nello stesso momento in più cache, allora si rischia che le copie si disallineino
- Le CPU devono rispettare particolari protocolli per garantire la coerenza

Shared vs. Distributed Memory

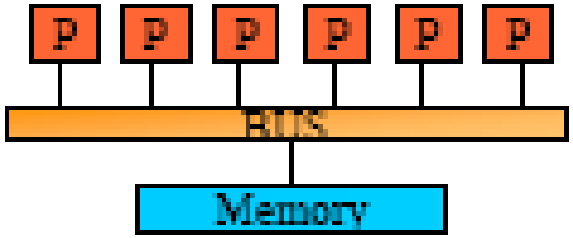


Shared memory - single address space. All processors have access to a pool of shared memory. (Ex: SGI Origin, Sun E10000)

Distributed memory - each processor has it's own local memory. Must do message passing to exchange data between processors. (Ex: CRAY T3E, IBM SP, clusters)

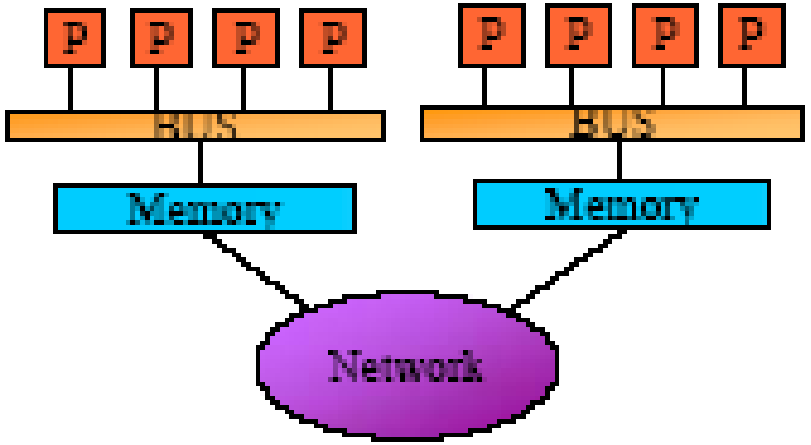


Shared Memory: UMA vs. NUMA



Uniform memory access (UMA): Each processor has uniform access to memory. Also known as symmetric multiprocessors (Sun E10000)

Non-uniform memory access (NUMA): Time for memory access depends on location of data. Local access is faster than non-local access. Easier to scale than SMPs (SGI Origin)



Linguaggi di Programmazione

- **HPF** (High Performance Fortran)
  - Viene usato per applicazioni data-parallel (suddivisione di dati tra i processi)
- **MPI** (Message Passing Interface)
  - Libreria per C e Fortran, si ottiene il parallelismi tramite scambio di messaggi
- **OpenMP**
  - Utilizzato per architetture a memoria condivisa (in pratica tramite threads e comandi fork/join)