La Top 500

Flops

- Differenza tra sustained performance, e di picco (cenni a proposito dei metodi di ottimizzazione, il compilatore ed oltre)

- La valutazione dell'effettiva potenza di calcolo dev'essere effettuata in relazione ad un benchmark standard, che consenta di comparare i valori ottenuti con quelli di altri elaboratori. Un riferimento in questo senso sono il Linpack.

- Cenni a proposito della TOP500 http://www.top500.org/

Top500 - Cray Jaguar

- AMD x86_64 Opteron Six Core 2600 MHz
- **1.75** petaflops (peak 2.33 petaflops)
- 224256 cores

Top500 - Nebulae

- Intel X5650, NVidia Tesla C2050 GPU
- **1.27** petaflops (peak 2.98 petaflops)
- National Supercomputing Centre in Shenzhen (NSCS)

4640 nodi ogni nodo ha due processori X5650 ed una GPU C2050, per un totale di **9280 processori** e **4640 GPUs** (un totale complessivo di ***120640*** *cores*, considerando i 6 cores a processore ed le 14 unita' SIMD per GPU)

Top500 - RoadRunner

Los Alamos National Laboratory nel New Mexico

1,03 petaflops Il sistema e' un computer ibrido basato su 7000 processori AMD Opteron e su 13000 processori PowerCell 8i processori derivati dal processore Cell.

Top500 - RoadRunner

AMD Opteron 2210, 1.8 GHz. Ogni processore ha due cores. Questi processori sono usati sia nella computazione (essenzialmente fungono da master node per i processori Cell), sia nella operazione di sistema. In totale il RoadRunner e' costituito da 6912 processori Opteron (6480 computation, 432 operation) per un totale di (12960+864) **13824 cores.**

IBM PowerXCell 8i, 3.2 GHz. Questi processori hanno un core general purpose (PPE), ed otto special performance cores (SPE) per le operazioni floating point. Roadrunner ha un totale di 12960 processori PowerXCell, con 12960 PPE cores e 103680 SPE cores, per un totale di **116640 cores.**

13824 Opteron cores + 116640 Cell cores = 130464 cores

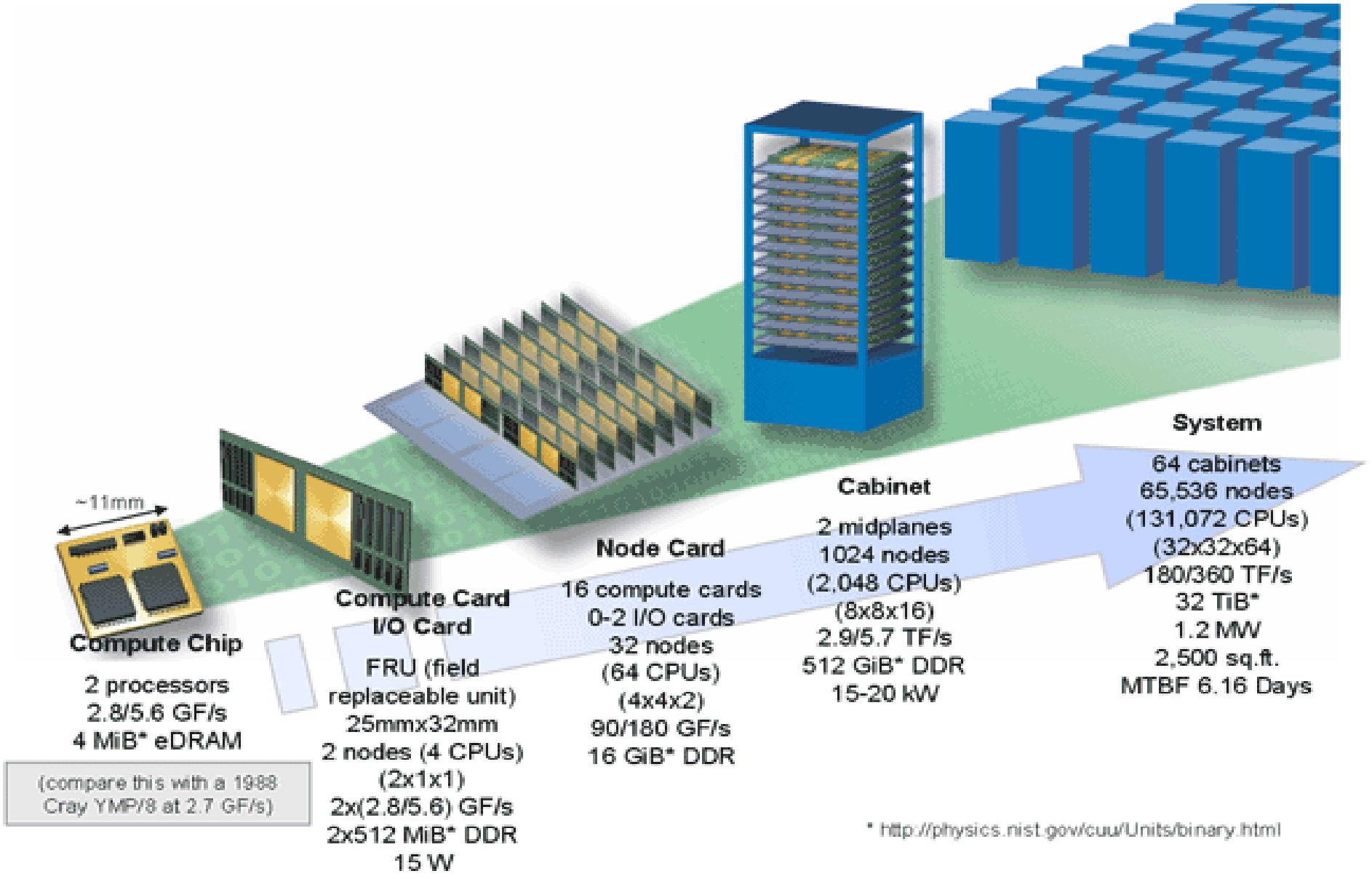
2.35 MW power

Top500 - BlueGene

DOE/NNSA/LLNL United States BlueGene/L - eServer Blue Gene Solution IBM

280,6 teraflops (Linpack) con una configurazione formata da **65.536 nodi di calcolo** e **1024 nodi di I/O.**

Top500 - BlueGene



Blue Gene - Architecture

- Each Compute or IO node integrates two 700 MHz PowerPC 440 embedded processors (each with a double-pipeline-double-precision Floating Point Unit (FPU)), with associated DRAM memory chips (512 MB)
- Each BlueGene/L node has a theoretical peak performance of 5.6 GFLOPS and dissipates low power. (17 watts)

Blue Gene - Network

Each Blue Gene/L node is attached to three parallel communications networks:

- a 3D toroidal network for peer-to-peer communication between compute nodes
- a collective network for collective communication
- a global interrupt network for fast barriers.

The I/O nodes (which run the Linux) provide communication with the world via an Ethernet network.

A separate and private Ethernet network provides access to any node for configuration and diagnostics.

Blue Gene - Software

- Blue Gene/L Compute nodes use a **minimal operating system** supporting a single user program. To allow multiple programs to run concurrently, a Blue Gene system can be partitioned into electronically isolated sets of nodes.
- I/O nodes handle the I/O needs of the compute nodes, and in some cases perform significant management activities, such as booting the compute node kernels or handling system calls not supported by the compute nodes.
- With so many nodes, component failures are inevitable. The system is able to electrically isolate faulty hardware to allow the machine to continue to run.

(PetaFLOPS = 10^{15} FLoating-point Operations Per Second)

Stime potenza di calcolo mente umana 20-100 petaflops ?

Tre esempi

- HP cluster
- **23 bi-processors** nodes Intel(R) Pentium(R) D CPU 3.00GHz
- Central Memory: 4GB
- **4 bi-processors** nodes Intel(R) Xeon(R) CPU 3.00GHz
- Central Memory: 16GB on each node
- Peak performance 3**50 Gflops**

- **Intel Itanium2 Beowulf Cluster (2003)**
  - 8 biprocessor nodes
  - **16 CPU** Intel Itanium2 (1.0/1.3Ghz)
  - 4Gb RAM for node (**32Gb** installed memory)
  - **70 Gflops** peak performance
  - **1152Gb** (**1Tb**) scratch disk space
  - Linux RedHat Operating System (IA64 64bits)
  - FastEthernet copper switched networking

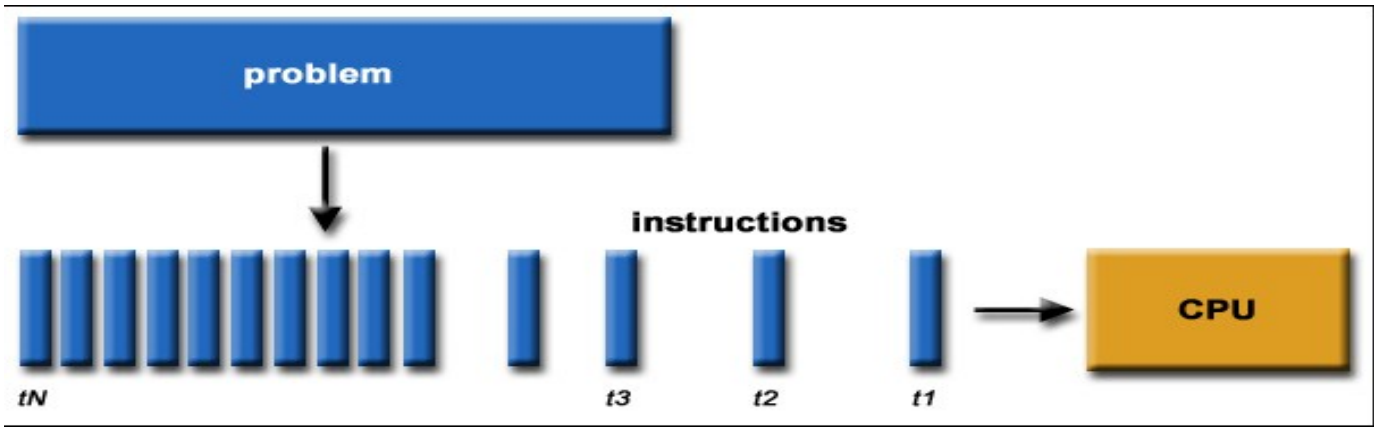
- **Intel Pentium4 Beowulf Cluster (ChemGRID 2004)**
  - 8 single processor nodes
  - **8 CPU** Intel P4 3.06Ghz
  - 1Gb RAM for node (**8Gb** installed memory)
  - **20 Gflops** peak performance
  - **576Gb** scratch disk space
  - Linux RedHat Operating System (IA32 32bits)
  - Gigabit Ethernet copper switched networking

Il calcolo parallelo

Calcolo sequenziale

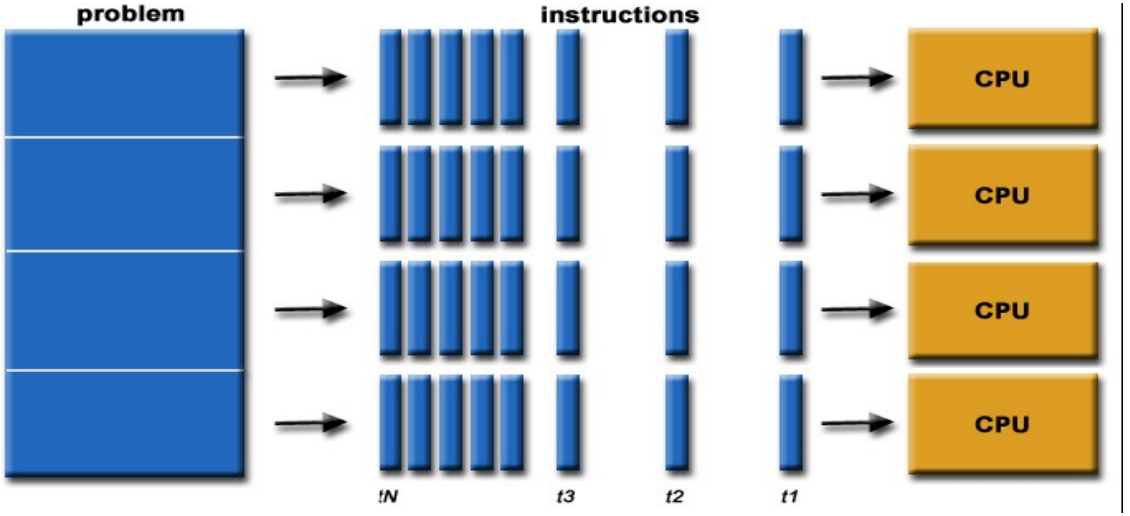
Tradizionalmente il software e' stato scritto per la computazione seriale, ovvero per girare su un singolo computer dotato di una singola CPU

- Un problema e' diviso in una serie discreta di istruzioni
- Le istruzioni sono eseguite una dopo l'altra (almeno in teoria!)
- In un certo istante e' eseguita una sola istruzione (almeno in teoria!)



Calcolo parallelo

- In generale, il calcolo parallelo e' l'uso simultaneo di piu' computer per risolvere un problema computazionale
- Per girare su piu' CPU, un problema e' diviso in parti discrete che possono essere risolte concorrentemente
- Ogni parte e' a sua volta divisa in una serie di istruzioni



- Le istruzioni di ogni parte sono eseguite contemporaneamente su CPU diverse (non necessariamente CPU diverse)

Perche' il calcolo parallelo ?

I principali motivi per usare il calcolo parallelo:

- Risparmiare tempo (minore wall clock time)
- Risolvere problemi piu' grandi
- Utilizzare un insieme di risorse non locali, disponibili su WAN o anche su Internet
- Contenere i costi: utilizzare contemporaneamente molte risorse computazionali economiche piuttosto che un singolo supercomputer costoso
- Superare vincoli di memoria
  - Un singolo computer possiede risorse di memoria finite
  - Per i problemi piu' grandi, utilizzare la memoria di piu' computer puo' aggirare questo vincolo

Grand Challange Problems

Storicamente il calcolo parallelo e' stato motivato dalla simulazione numerica di sistemi molto complessi:

- Understanding matter from elementary particles (quantum chemistry) to cosmology
- Storm forecasting and climate prediction
- Understanding biochemical processes of living organisms
- Drug design
- Nuclear weapons stewardship
- Advanced vehicle design
- Pollution modeling and remediation planning
- Molecular nanotechnology
- Cryptology
- Data mining
- Information retrieval
- Computational finance

Limiti del computing seriale

Ci sono ragioni fisiche e pratiche che pongono un serio vincolo alla costruzione di computer seriali sempre piu' veloci:

- Limiti alla velocita' di trasmissione.
  - La velocita' di un computer seriale dipendente da quanto i dati possono essere spostati velocemente nell'hardware
  - Il limite assoluto e' la velocita' della luce (30 cm/nsec) ed il limite di trasmissione in un filo di rame (9 cm/nsec)
  - Per incrementare la velocita' di trasmissione e' necessario aumentare la prossimita' dei processing elements
- Limiti alla miniaturizzazione
  - La tecnologia dei processori permette di inserire un numero sempre piu' alto di transistor in un chip
  - Anche con componenti a livello atomico/molecolare un limite di miniaturizzazione sara' comunque raggiunto

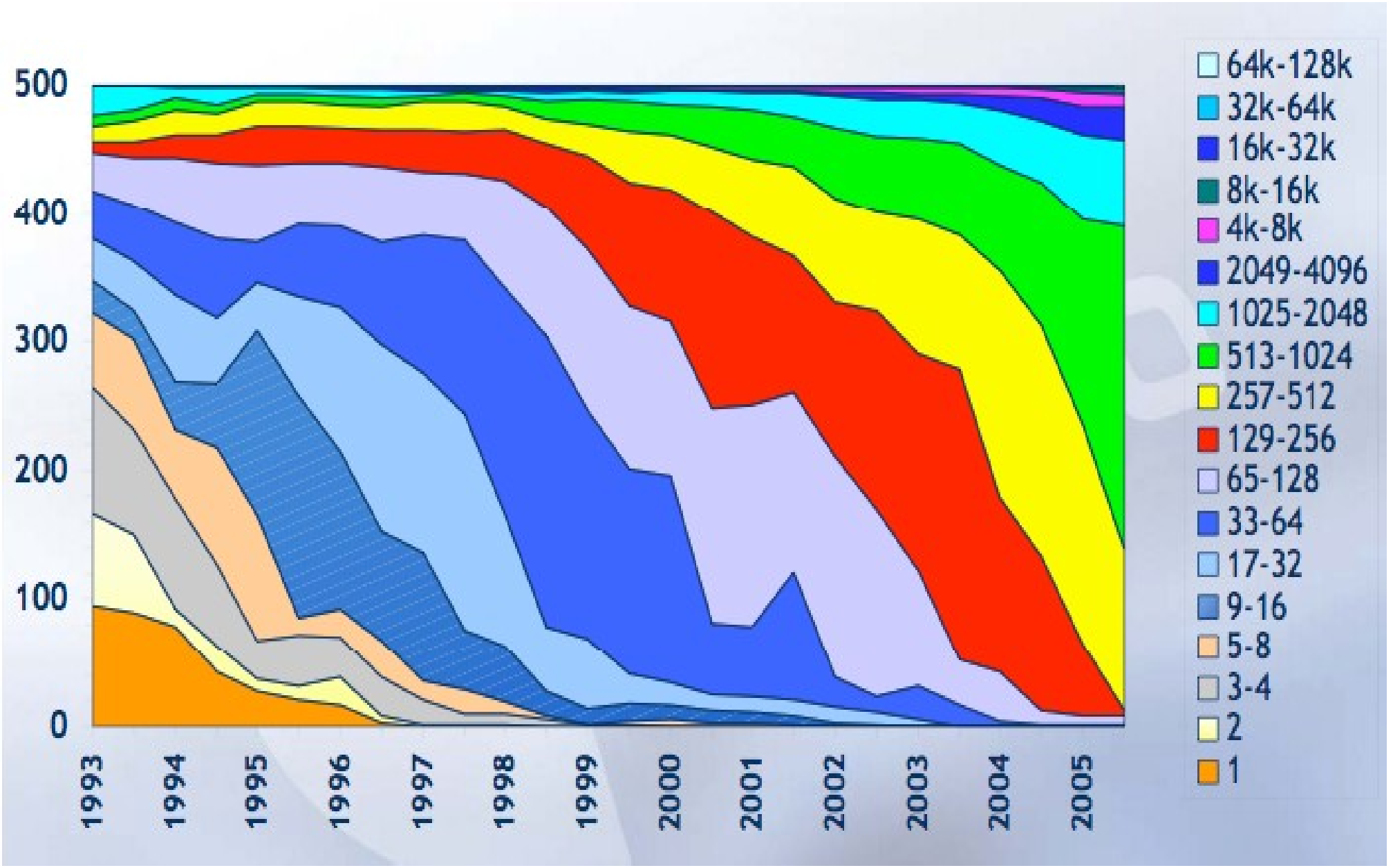
Limiti economici

- E' sempre piu' costoso realizzare processori sempre piu' veloci
- E' meno costoso utilizzare un numero piu' grande di processori moderatamente veloci per raggiungere la stessa performance, o addirittura una performance piu' alta
- Negli ultimi 10 anni i trend indicano l'imporsi di
  - network sempre piu' veloci
  - sistemi distribuiti
  - architetture di computer multi processore, anche a livello di desktop e laptop

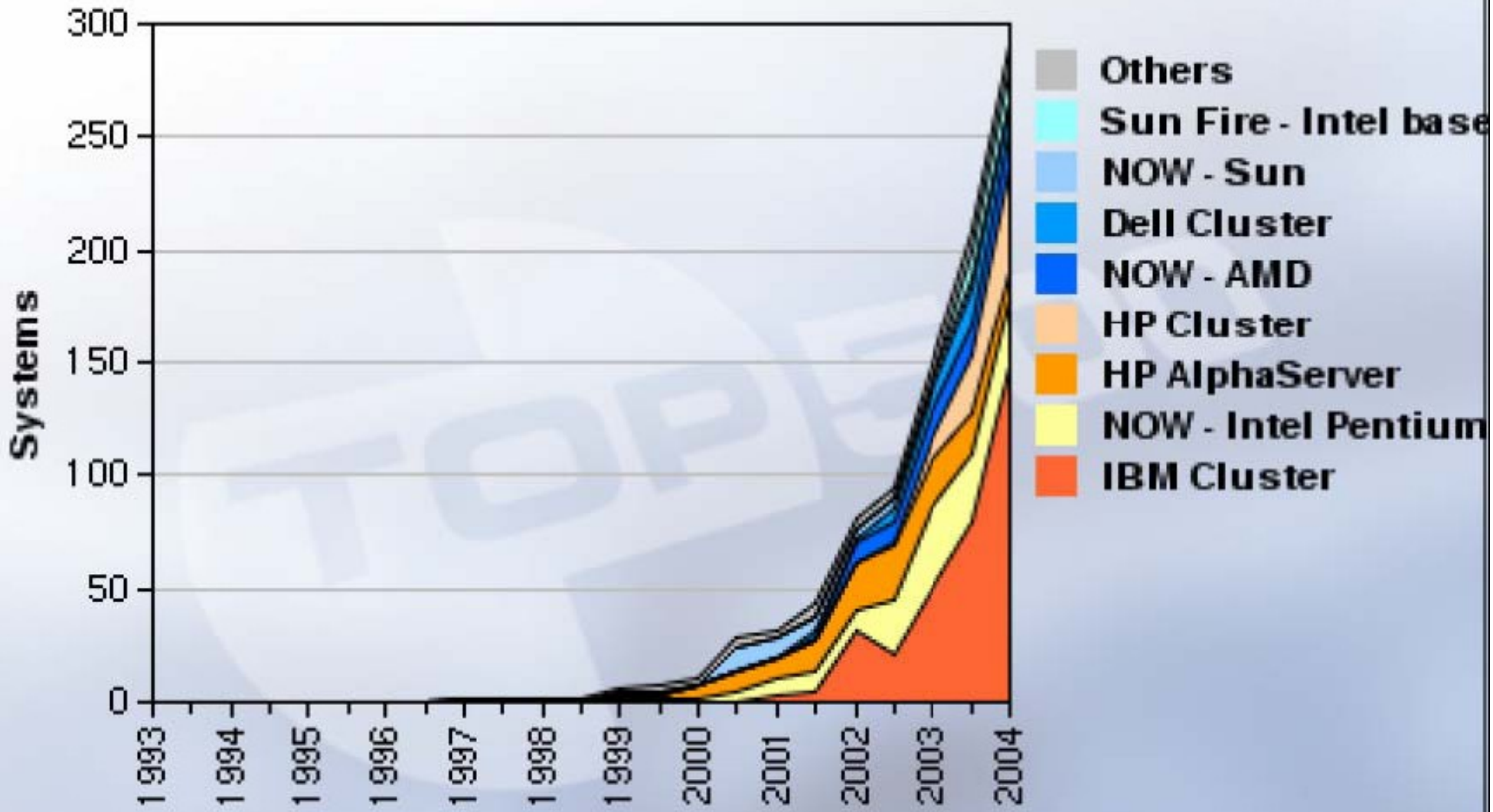
Il parallelismo e' il futuro del computing ?

Il Parallelismo Oggi

- Parallelismo implicito
  - pipeline (esecuzione pipelined implicita delle istruzioni)
- Parallelismo esplicito
  - istruzioni multimediali e di streaming (MMX, SSE vedi opzioni del compilatore)
- Processori multicore
  - AMD: Opteron dual core (caches separate)
  - Intel: Core 2 Duo, dual core (cache condivisa)
  - Sun: Niagara, 8 cores
  - IBM: Power 5, dual core; Cell 1 PPC core ed 8 SPE
- Cluster (Beowulf cluster)
- SuperComputer (vedi il trend della top500)

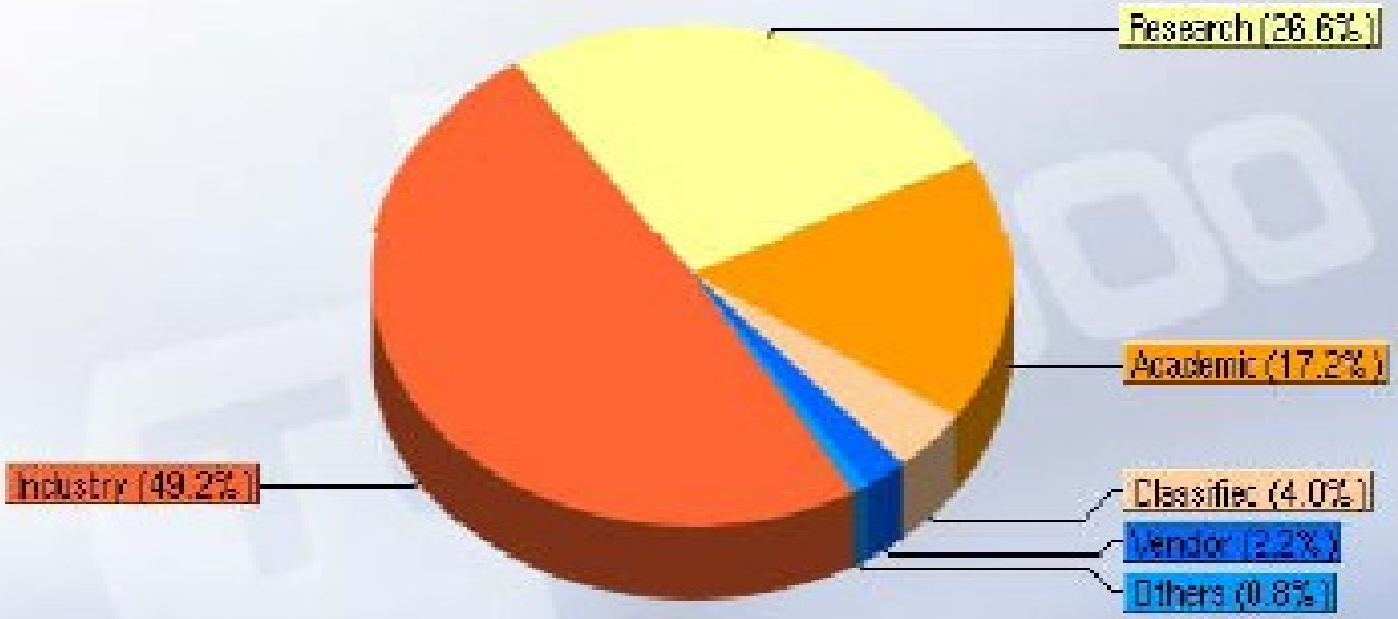

500
400
300
200
100
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
64k-128k
32k-64k
16k-32k
8k-16k
4k-8k
2049-4096
1025-2048
513-1024
257-512
129-256
65-128
33-64
17-32
9-16
5-8
3-4
2
1

Clusters (NOW) / Systems



Customer Segment / Systems (November 2006)

November 2006

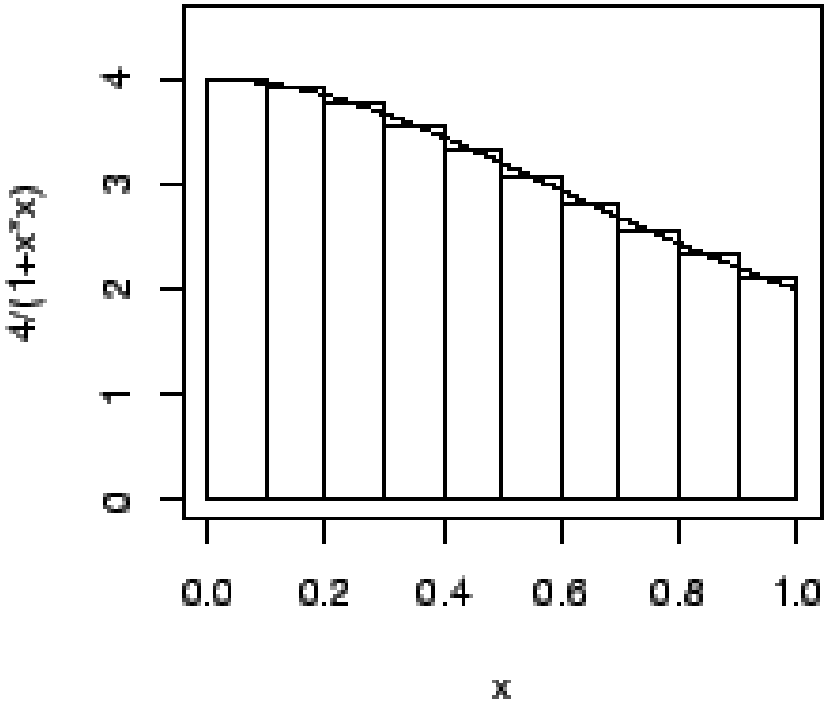


Calcolo di PI

Calcolo PI

$$\pi = \int_0^1 \frac{4}{1+x^2}\mathrm{d}x \sim \frac{1}{n}\sum_{i=1}^{n} \frac{4}{1+\left(\frac{i-0.5}{n}\right)^2}.$$

Per n = 10 vedi la figura:



Provate ad implementarlo

Calcolo PI

```
int n, i;
double d, s, x, pi;

n = NUMOFP;
d = 1.0/n;
s = 0.0;

for (i=1; i<=n; i++){
  x = (i-0.5)*d;
  s += 4.0/(1.0+x*x);
}

pi = d*s;
```

Calcolo PI – Process forking

Un processo e' un cotesto (un'entita') nel quale viene seguito un determinato codice.

Ogni processo ha il proprio stack, le proprie pagine di memoria, la propria file descriptors table, un proprio PID. (due parole su come e' implemtato il multi-tasking).

Proveremo a realizzare un prima parallelizzazione dell'algoritmo visto sopra usando fork(), wait() ed meccanismi base di IPC (Inter-Process Comunication).

Calcolo PI – Process forking

```
$ time ./forking 100000000
computed pi value = 3.14159265359002226603025...
                    3.14159265358979323846264...

real    0m1.472s
user    0m2.872s
sys     0m0.006s

$ time ./seq 100000000
computed pi value = 3.14159265359042638721121...
                    3.14159265358979323846264...

real    0m2.836s
user    0m2.830s
sys     0m0.006s
```

Calcolo PI – Process forking

Qualche instabilita' numerica (ovvia) a parte il risultato e' evidente.

In generale non e' pero' conveniente usare il forking per via delle latenze dovute alla copia delle aree di memoria durante la creazione dei processi.

Esercizio: provate a fare la somma dei primi n numeri naturali, sequenziale ed usando il process forking

Somma Primi N naturali – Process forking

```
$ time ./summa 20000001
result = 200000030000001.000000

real    0m0.076s
user    0m0.076s
sys     0m0.000s


$ time ./forking 20000001
result = 200000030000001.000000

real    0m0.052s
user    0m0.076s
sys     0m0.001s
```

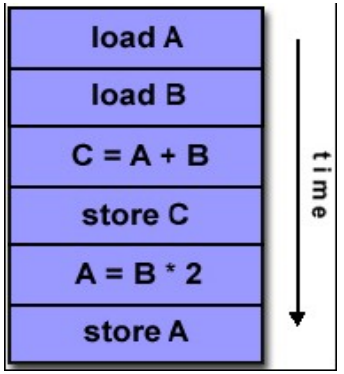
Tassonomia di Flynn

Ci sono diversi modi per classificare i computer paralleli, una delle classificazioni storicamente più usate è nota come tassonomia di Flynn ed e' usata fin dagli anni '60:

- La tassonomia di Flynn distingue le architetture multi processore secondo due parametri indipendenti: Istruzioni e Dati.
- Ognuno di questi parametri può avere due stati: Single o Multiple:
  - **MIMD** Multiple Instruction, Multiple Data
  - **MISD** Multiple Instruction, Single Data
  - **SIMD** Single Instruction, Multiple Data
  - **SISD** Single Instruction, Single Data

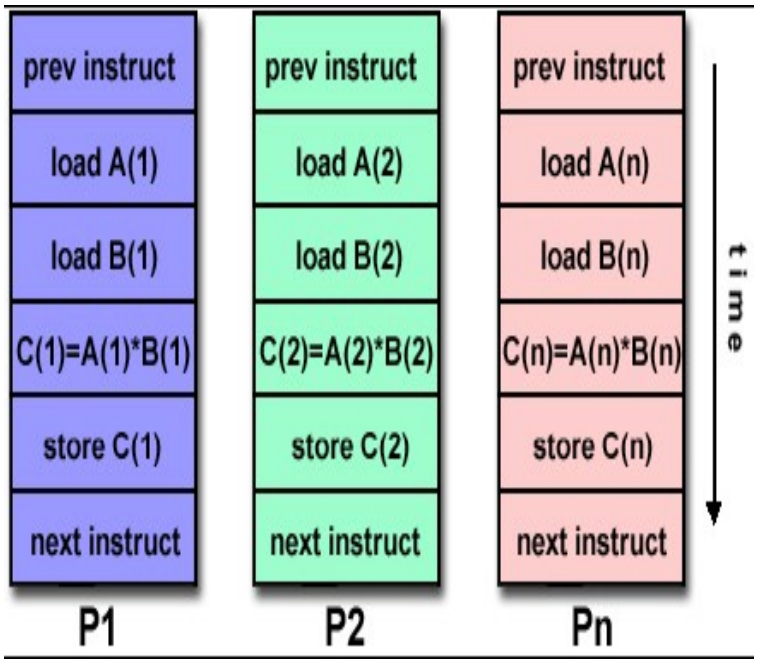
SISD

- Un computer seriale
- Single instruction: solo un flusso di istruzioni e' attivo nella CPU in ogni ciclo di clock
- Single data: un flusso di dati puo' essere usato come input durante ogni ciclo di clock
- L'architettura prevalente dei computer
- Esempi: i PC ordinari, workstation a singola CPU e tradizionali mainframe



SIMD

- Single instruction: tutti i processori ese
  la stessa istruzione ad ogni ciclo di cloc
- Multiple data: ogni processore opera su
  dati diversi



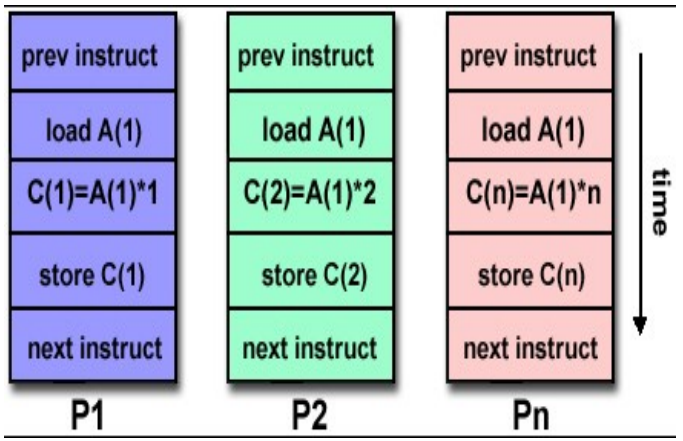
SIMD

- Tipicamente implementata con un nodo controller (responsabile del flusso d'istruzioni), una grande array di unità di calcolo di capacita' medio piccola ed un network d'interconnessione veloce
- Si adatta a specifici problemi, con un alto grado di regolarita' (es. image processing)

- Esempi:
  - Le macchine APE, progettate e realizzate dall'INFN (commercializzata poi dalla societa' Quadrics. Linguaggio di programmazione TAO che e' una versione estesa del Fortran)
  - I processori multimediali (mostrano una forma limitata di parallelismo di tipo SIMD)
  - I calcolatori vettoriali

MISD

- Un singolo flusso di dati e' elaborato da piu' unita' di processing.
- Ogni unita' di processing opera sui dati indipendentemente attraverso un proprio flusso di istruzioni

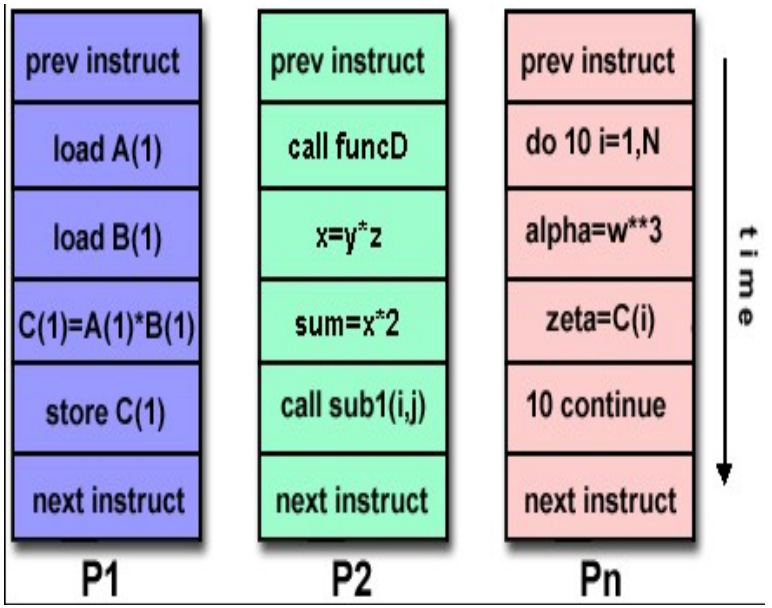


MISD

- Sono esistiti pochissimi esempi di questo tipo d'architettura, come il computer sperimentale Carnegie-Mellon C.mmp (1971)
- Usi concepibili:
  - Filtri multi frequenza da applicare su un singolo segnale di input
  - Applicazione di diversi algoritmi per decrittare lo stesso set di dati codificato
- Architetture tipo Pipeline possono essere considerate MISD (anche se ad ogni stage della pipeline il dato e' differente)

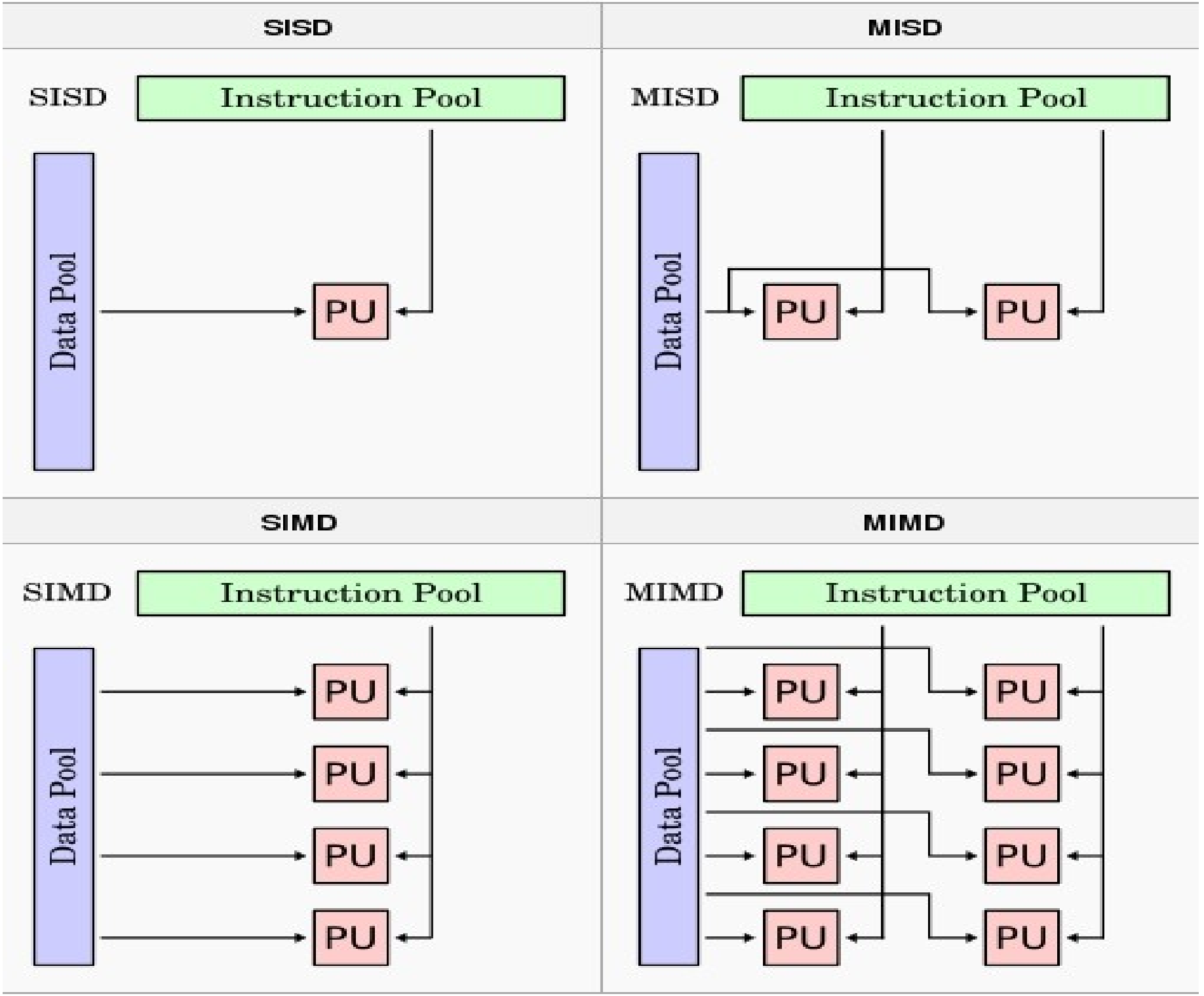
MIMD

- L'architettura di computer parallelo piu' diffusa
- I piu' moderni elaboratori paralleli sono disegnati secondo questa architettura



MIMD

- Multiple Istruction: ogni processore puo' eseguire un flusso di istruzioni diverso
- Multiple Data: ogni processore puo' lavorare su un flusso di dati diverso
- L'esecuzione puo' essere sincrona o asincrona
- Esempi:
  - molti degli attuali supercomputer, i cluster, …


SISD
SISD
Instruction Pool
Data Pool
PU
MISD
MISD
Instruction Pool
Data Pool
PU
PU
SIMD
SIMD
Instruction Pool
Data Pool
PU
PU
PU
PU
MIMD
MIMD
Instruction Pool
Data Pool
PU
PU
PU
PU
PU
PU
PU
PU