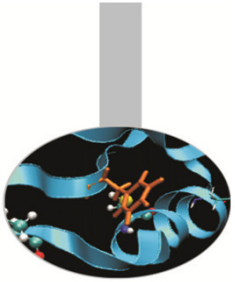
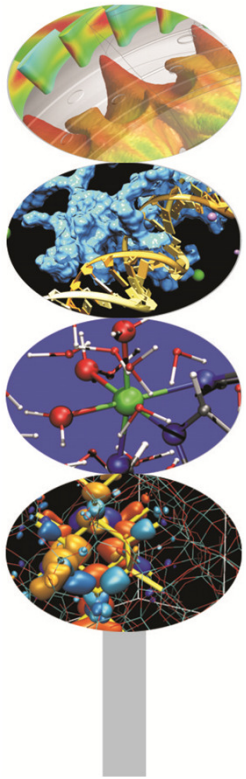
CINECA SCAI
SuperComputing Applications and Innovation



Architetture



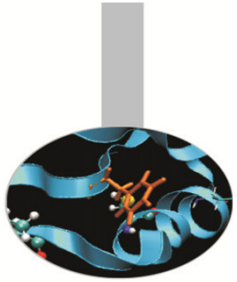
Paride Dagna

SuperComputing Applications and Innovation Department

18/02/2013

CINECA

Introduzione

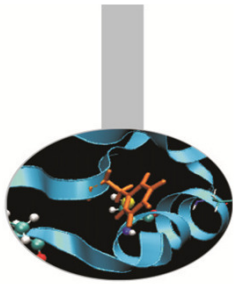


Grazie alle moderne tecniche di programmazione e agli strumenti di sviluppo attualmente disponibili, risulta piuttosto semplice scrivere applicazioni seriali o parallele, anche per il programmatore meno esperto.

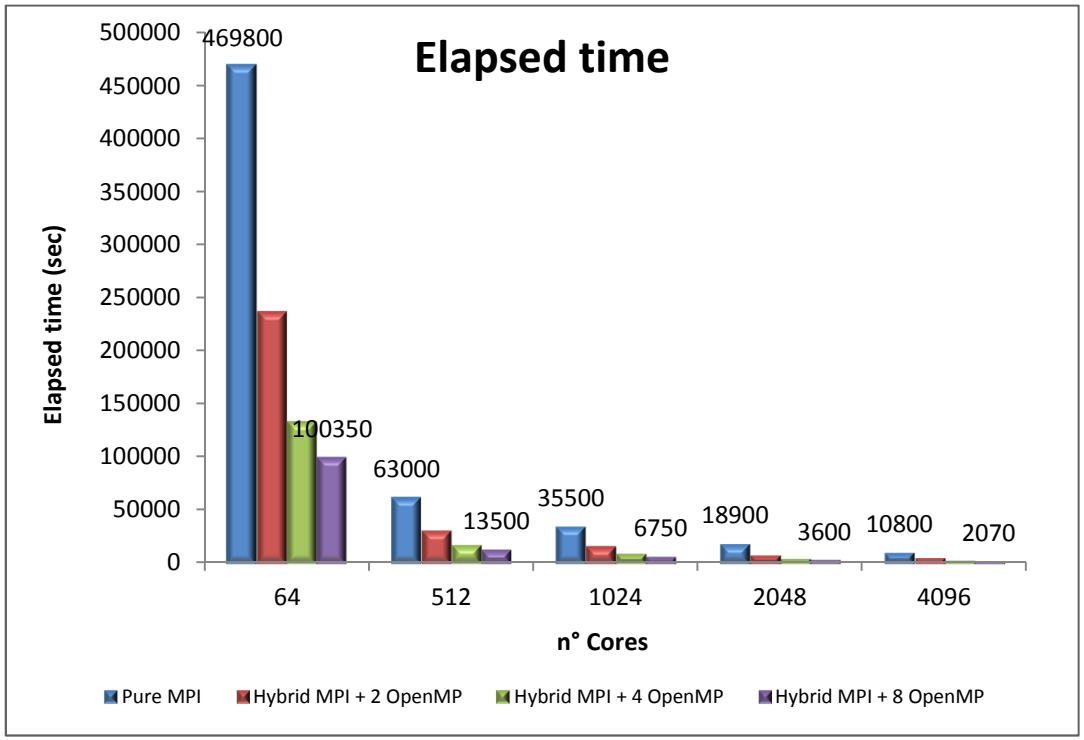
Il processo di ottimizzazione del codice è invece un percorso molto più complesso che richiede esperienza e una buona conoscenza dell'architettura dei sistemi disponibili.

In fase di progettazione del modello, conoscere le caratteristiche e le risorse hardware dei sistemi (cpu, memorie, rete di interconnessione) è fondamentale per riuscire a sfruttare in modo ottimale tutte le potenzialità delle moderne architetture.

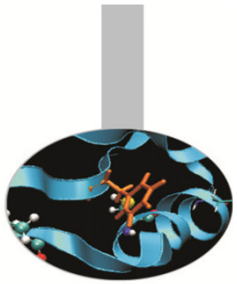
Introduzione



- Codice agli elementi spettrali per la simulazione tridimensionale di eventi sismici su larga scala
  - ➢ Un'attenta revisione del software nella parte di calcolo, comunicazione fra i processi e I/O ha permesso di ottenere su Fermi (Blue Gene/Q), a parità di cores, un incremento medio delle performance di 5-6 volte

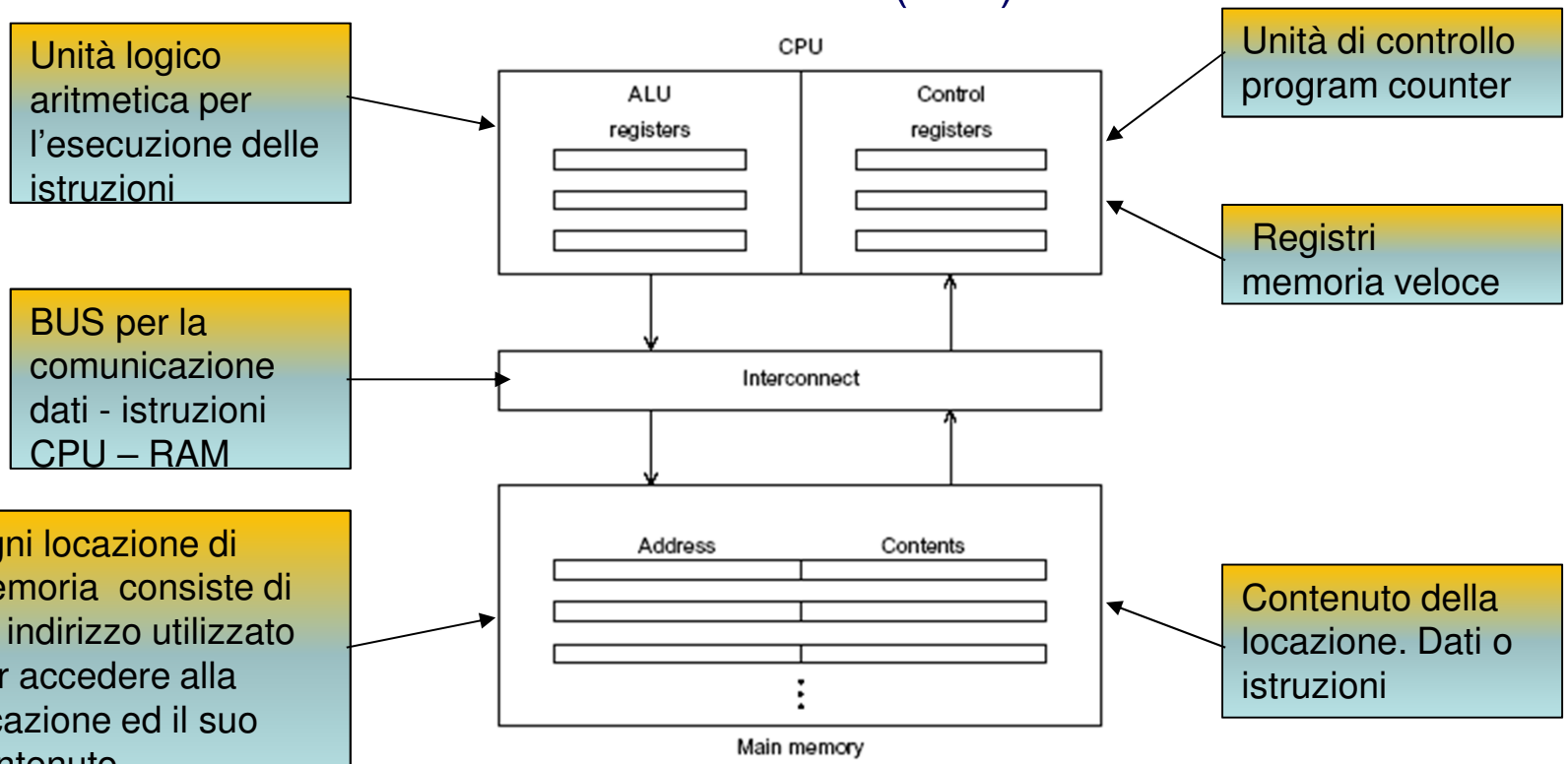


Architettura di von Neumann

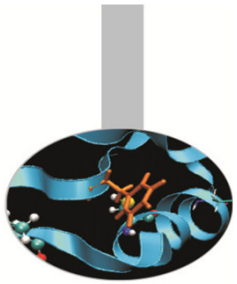


Costituita da:

- Central processing unit (CPU)
- Random Access Memory (RAM)
- Interconnessione CPU – RAM (Bus)



Architettura di von Neumann



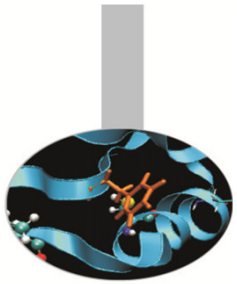
- I dati sono trasferiti dalla memoria alla CPU (**fetch o read**)

- I dati sono trasferiti dalla CPU alla memoria (**written to memory o stored**)

- La separazione di CPU e memoria è conosciuta come la «**von Neumann bottleneck**» perché è il **bus** di interconnessione che determina a che velocità si può accedere ai dati e alle istruzioni.

- Le moderne CPU sono in grado di eseguire istruzioni almeno cento volte più velocemente rispetto al tempo richiesto per recuperare (**fetch**) i dati in RAM

Caching - Virtual Memory - Instruction Level Parallelism



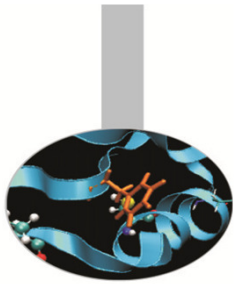
- L'evoluzione dei sistemi e dei microprocessori e l'ingegnerizzazione dei calcolatori ha affrontato il «**von Neumann bottleneck**» seguendo tre percorsi paralleli:

- **Caching**
  - Memorie molto veloci presenti sul chip del processore. Esistono cache di primo, secondo e terzo livello.

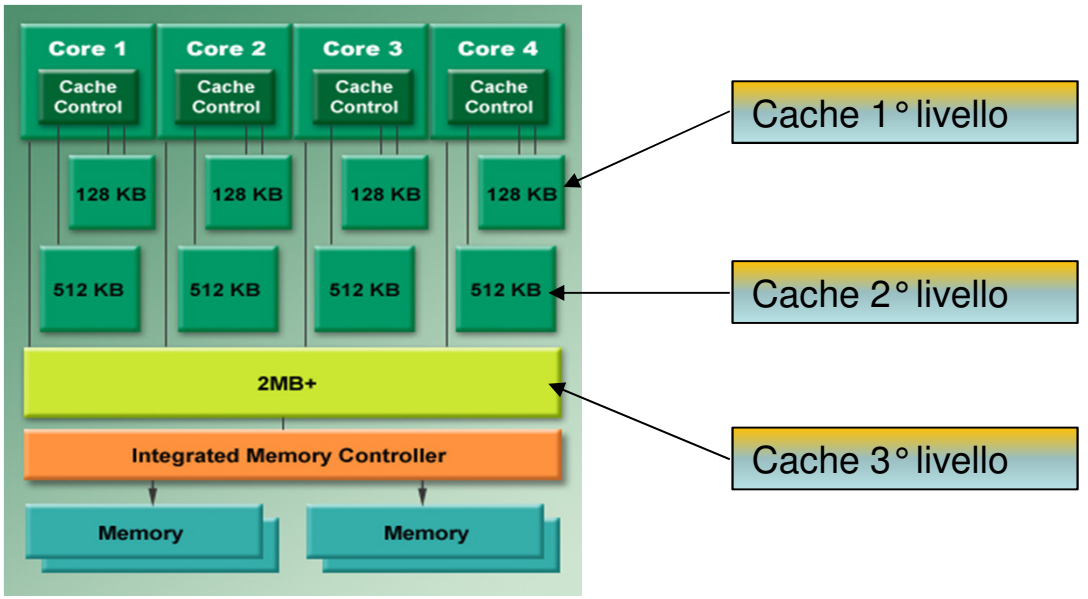
- **Virtual memory**
  - Sistema sviluppato per fare in modo che la RAM funzioni come una cache per lo storage di grosse moli di dati.

- **Instruction level parallelism**
  - Tecnica utilizzata per avere più unità funzionali nella CPU che eseguono istruzioni in parallelo (**pipelining – multiple issue**)

Caching

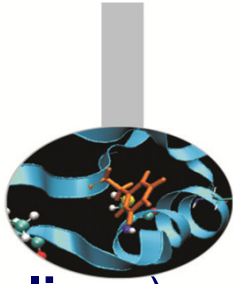


- La cache è una collezione di locazioni di memoria che possono essere accedute dalla CPU in pochissimi cicli di clock.
  Nelle moderne soluzioni hardware le cache risiedono nello stesso chip della CPU.



- La memorizzazione di dati e istruzioni in cache segue i due principi universalmente accettati di località spaziale e temporale, basati sull'idea che i programmi tendano ad utilizzare dati ed istruzioni che sono fisicamente vicini ai dati e alle istruzioni appena eseguite.

Caching



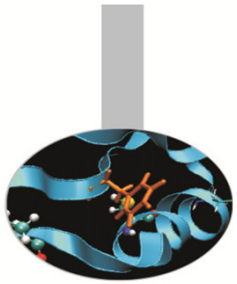
- Dati e istruzioni sono memorizzati in blocchi (**cache block o cache lines**) e non singolarmente.
  Tipicamente dati e istruzioni vengono letti e memorizzati sulle linee di cache a gruppi di 8 o 16.

fortran
```
REAL,DIMENSION(1000) :: A
REAL :: sum=0.0
DO I = 1, 1000
              sum = sum + A(I)
END DO
```

c/c++
```
float a[1000];
int sum =0.0;
for(i=0; i<1000; i++){
         sum+=a[i];
}
```

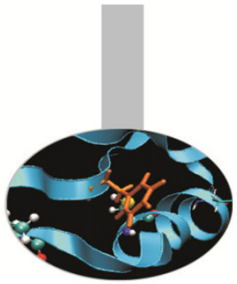
- Questa tecnica incrementa notevolmente le prestazioni del calcolo dal momento che la CPU trova subito in cache dati e istruzioni senza la necessità di doverli reperire in RAM e quindi dover passare per il collo di bottiglia del BUS.
- Nei moderni processori a 2 o 3 livelli di cache le informazioni vengono cercate in modo gerarchico dal primo livello fino al terzo (L1 → L2 → L3)

Caching

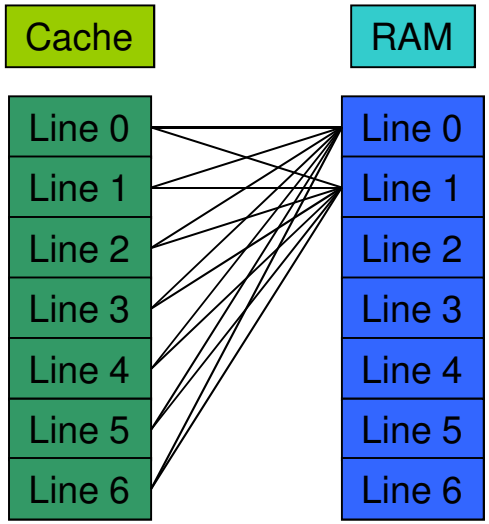


- Cache **fully associative**
  - ogni linea di dati o istruzioni presa dalla RAM può essere memorizzata in una qualsiasi linea della cache
- Cache **n-way set associative**
  - ogni linea di dati o istruzioni presa dalla RAM può essere memorizzata in una qualsiasi delle **n** differenti locazioni
- Cache **direct mapped**
  - ogni linea di dati o istruzioni presa dalla RAM può essere memorizzata in un'unica linea di cache

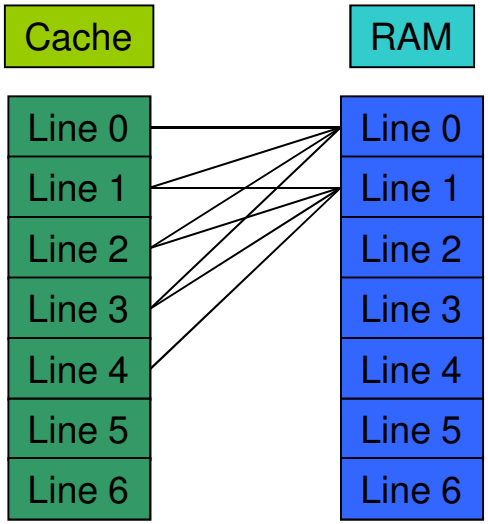
Caching



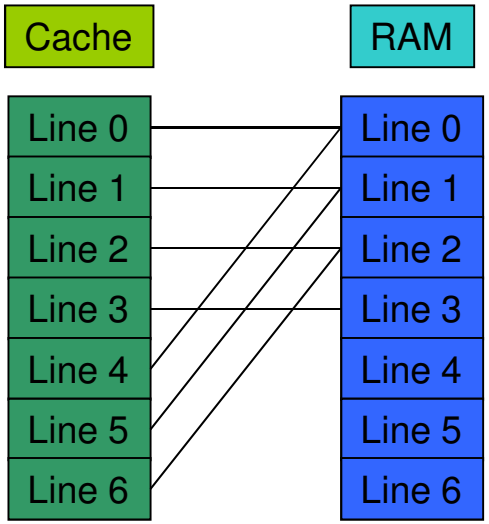
Cache
fully associative



Cache
n-way set associative

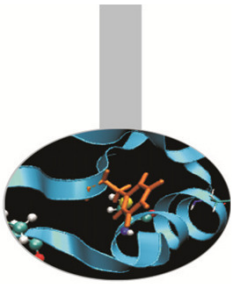


Cache
direct mapped

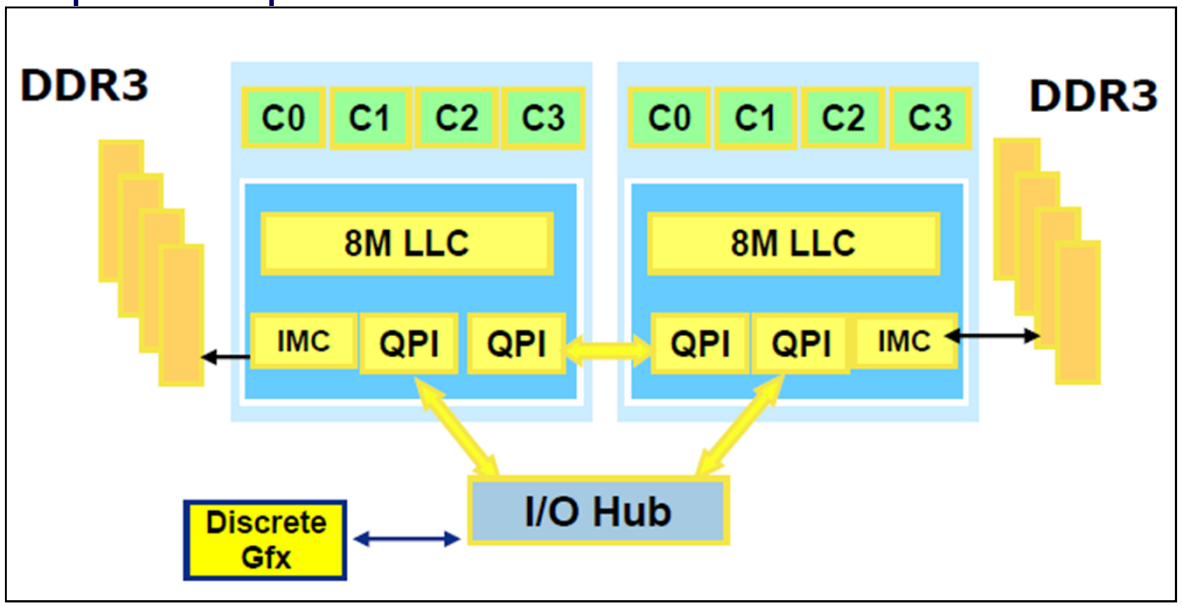


- Cache **fully associative**: ogni blocco di RAM può essere mappato su una linea qualsiasi di cache
- Cache **n-way associative**: ogni blocco di RAM può essere mappato su n linee di cache
- Cache **direct mapped**: ogni blocco di RAM può essere mappato su una sola linea di cache

Cache e RAM: costo accesso

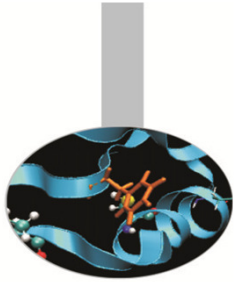


Tempi stimati per una piattaforma 2 socket Intel® Core™ i7 Processor



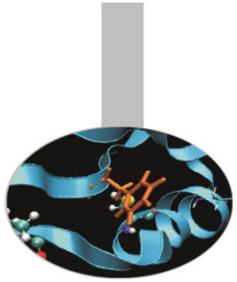
Livello	Costo di accesso
L1	4 cicli di clock
L2	10 cicli di clock
L3 condivisa sullo stesso socket	40 - 75 cicli di clock
L3 condivisa su altro socket	100 - 300 cicli di clock
RAM	60 - 100 ns

Virtual memory



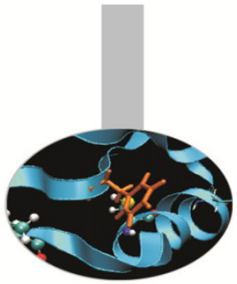
- Le caches permettono di accedere velocemente a dati e istruzioni che risiedono sulla RAM.
  Quando si utilizzano grandi data sets o programmi molto complessi può accadere che non tutti i dati e le istruzioni possano essere contenute in RAM. Questo è un evento abbastanza frequente negli attuali sistemi operativi che supportano l'esecuzione di molti processi contemporanei (multitasking).
- Le risorse di memoria sono condivise fra i vari processi che hanno la necessità di trovare disponibili dati e istruzioni.
- La **virtual memory** è stata sviluppata così che la RAM possa comportarsi come una cache per lo storage secondario (hard disk).
  Vengono possibilmente mantenute in RAM solo le risorse attive dei processi.
- Le parti inattive vengono invece mantenute nello spazio di **swap.**

Virtual memory



- Anche la memoria virtuale opera in blocchi chiamati **pages.**
- Dal momento che l'accesso allo storage secondario è centinaia di volte più lento rispetto alla RAM, la dimensione delle **pages** è relativamente elevata (4-16 kB).
- La **virtualizzazione delle pages** è necessaria per la gestione dinamica della memoria virtuale da assegnare ai processi nei sistemi multitasking.
- Quando un programma viene compilato, alle sue **pages** sono assegnati dei numeri di pagina virtuali.
- Quando viene lanciato, viene creata una **page table** che mappa i numeri di pagina virtuali a indirizzi fisici.
  L'effetto secondario è il raddoppio del tempo necessario per l'accesso ad una locazione di memoria reale in RAM ma in questo modo si è certi che processi diversi non cerchino di accedere agli stessi indirizzi fisici.
- L'indirizzo **virtuale** deve essere trasformato in un indirizzo di memoria fisico **reale**.

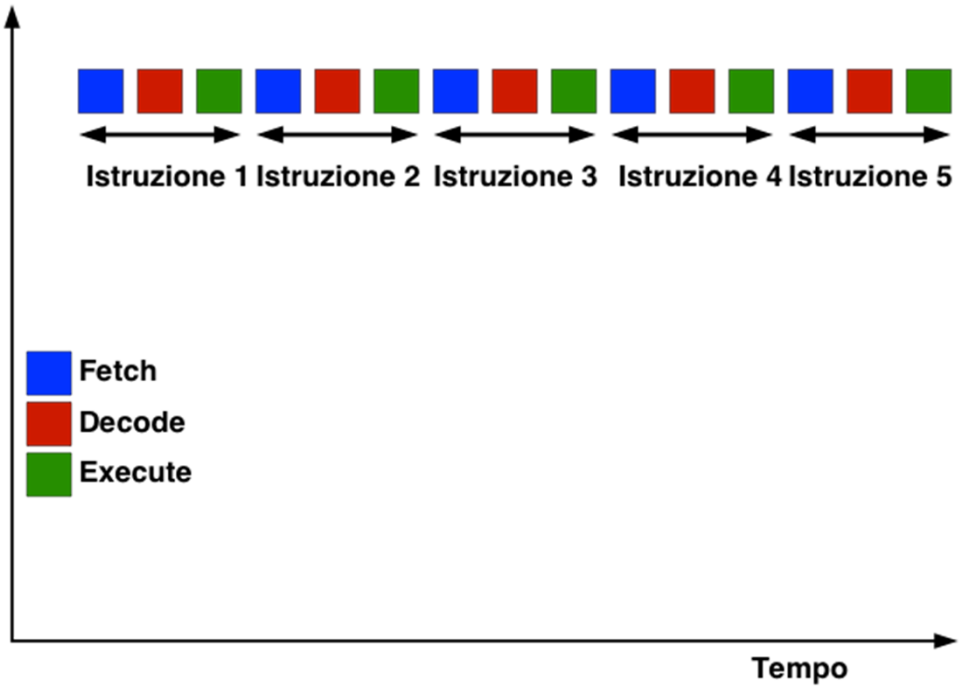
Instruction level parallelism



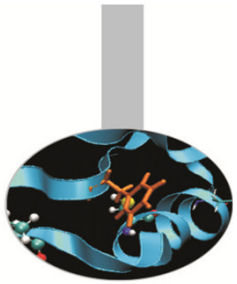
- Esempio: Una istruzione completata ogni tre cicli
- Spreco nell'utilizzo della logica

Dopo che l'istruzione è stata caricata ed avviata all'unità di esecuzione, la logica di fetch rimane inutilizzata fino al completamento dell'istruzione, così come, dopo che l'istruzione ha lasciato l'unità di esecuzione e si avvia alla fase di ritiro, l'unità di esecuzione attende senza produrre lavoro utile.

- 1000 MHz
- Latenza 3 Cicli
- MFlops (picco) = 330

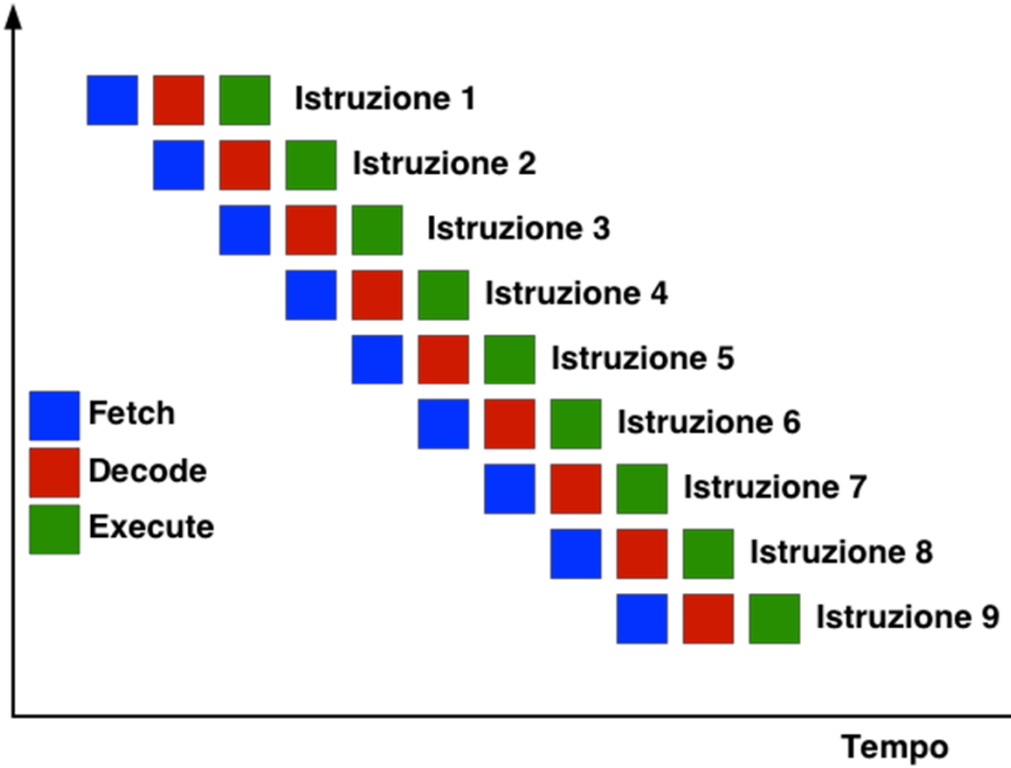


Instruction level parallelism

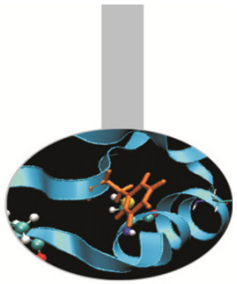


- Un risultato per ciclo a pipeline piena
  - Per riempirla servono 3 istruzioni indipendenti, con gli operandi
  - All'estremo opposto, un risultato ogni 3 cicli a pipeline "vuota"

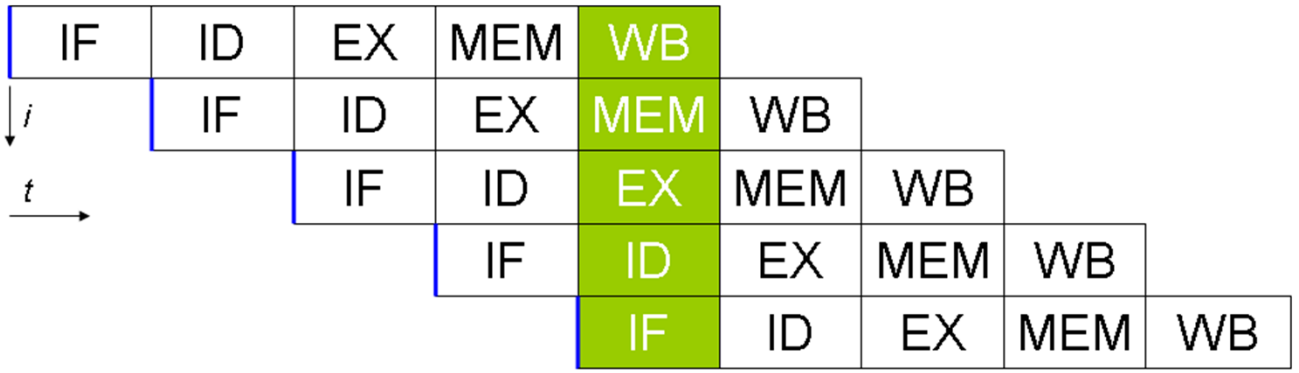
- 1000 MHz
- Latenza 3 Cicli
- 1 operazione FP/ciclo
- Mflops (picco) = 1000
- 3 volte più veloce
- 3 volte più BW



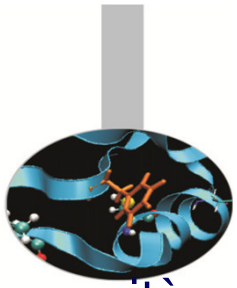
Instruction level parallelism



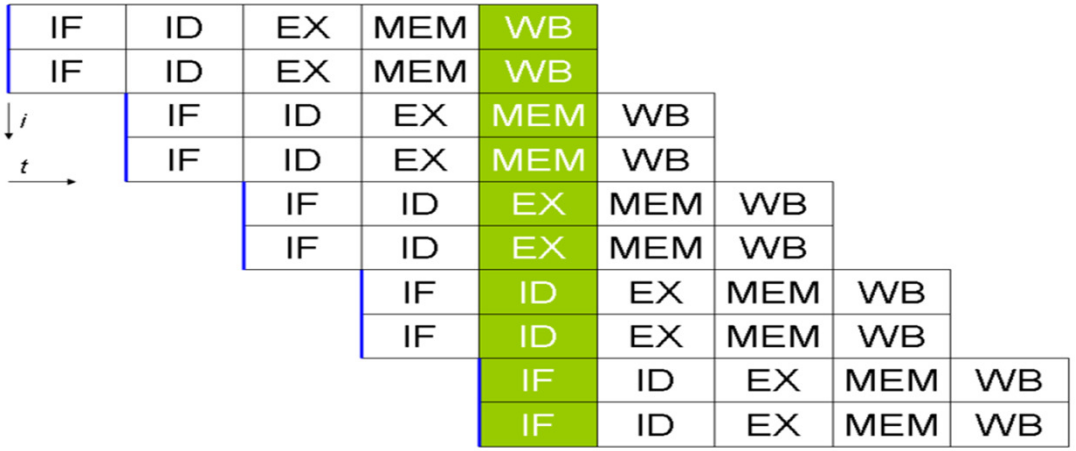
- **Instruction-level parallelism** (ILP), migliora le prestazioni del processore permettendo l'utilizzo contemporaneo delle diverse **unità funzionali** disponibili.
- Due possibili approcci per avere ILP sono:
  - **Pipelining,** dove le unità funzionali sono organizzate in stadi;
  - **multiple issue,** dove diverse istruzioni indipendenti possono essere inizializzate.
- La pipeline funziona come una catena di montaggio dove le varie unità funzionali possono lavorare in modo indipendente su dati diversi.



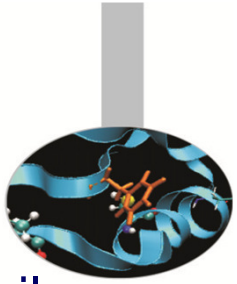
Instruction level parallelism



- La pipeline migliora le performance connettendo in sequenza diverse unità funzionali.
- Nei processori **multiple issue** le unità funzionali e anche le pipeline sono replicate per eseguire in parallelo istruzioni differenti all'interno di un programma o la stessa istruzione su dati diversi.
- Se le unità funzionali sono schedulate a compile-time il sistema di multiple issue è detto **statico**, se sono schedulate a run-time il sistema è detto **dinamico**.
- Un processore che supporta il multiple issue dinamico è detto **superscalare**.



Bandwidth

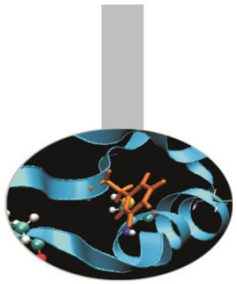


- La velocità con la quale è possibile trasferire i dati tra la memoria e il processore
- Si misura in numero di bytes che si possono traferire al secondo (Mb/s, Gb/s, etc..)

- A=B*C
  - leggere dalla memoria il dato B
  - leggere dalla memoria il dato C
  - calcolare il prodotto B * C
  - salvare il risultato in memoria, nella posizione della variabile A

- 1 operazione floating-point → 3 accessi in memoria

Stream

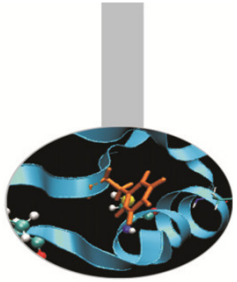


- **Benchmark per la misura della bandwidth da e per la CPU**

- Misura il tempo per le operazioni sottostanti fra gli array a,c,d e lo scalare b
  - Copia a → c (copy)
  - Copia a*b → c (scale)
  - Somma a+b → c (add)
  - Somma a+b*c → d (triad)
- http://www.cs.virginia.edu/stream/ref.html

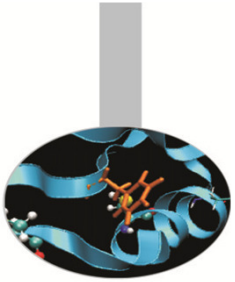
- **Compilazione** : digitare «make» nella cartella «architetture»
- **Opzioni compilazione** :
  - dimensione array : -DSTREAM_ARRAY_SIZE
  - Numero ripetizioni test : -DNTIMES
  - Versione multithread OpenMP : -fopenmp

Classificazione architetture - Flynn

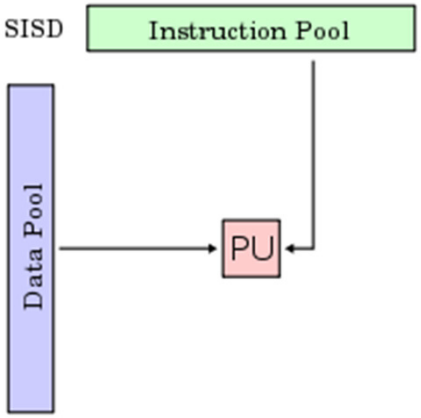


- La classificazione di Flynn categorizza un'architettura in base alla molteplicità dell'hardware usato per manipolare lo **stream** di istruzioni e dati.
- La tassonomia di Flynn racchiude le architetture dei computer in quattro categorie:
  - **SISD** : *single instruction, single data.* Corrisponde alla classica achitettura di von Neumann, sistema scalare monoprocessore.
  - **SIMD** : *single instruction, multiple data.* Architetture vettoriali, processori vettoriali, GPU.
  - **MISD** : *multiple instruction, single data.* Non esistono soluzioni hardware che sfruttino questa architettura.
  - **MIMD** : *multiple instruction, multiple data.* Più processori/cores interpretano istruzioni diverse e operano su dati diversi.
- Le moderne soluzioni di calcolo sono in realtà date da una combinazione delle categorie previste da Flynn.

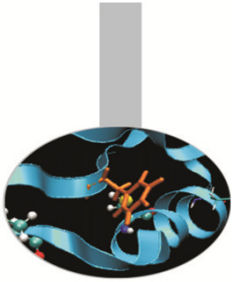
Classificazione architetture - SISD



- E' il **classico sistema di von Neumann** a cui appartengono i calcolatori con una sola unità esecutiva ed una sola memoria. Il singolo processore obbedisce ad un singolo flusso di istruzioni (programma sequenziale) ed esegue queste istruzioni ogni volta su un singolo flusso di dati.
- I **limiti** di prestazione di questa architettura vengono **ovviati** aumentando il **bus dati** ed i **livelli di memoria** ed introducendo un parallelismo attraverso le tecniche di **pipelining** e **multiple issue**.



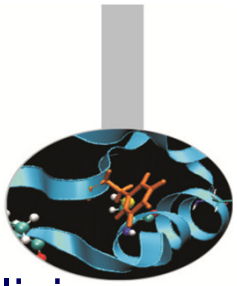
Classificazione architetture - SIMD



- La stessa istruzione viene eseguita in parallelo su dati differenti.
  - modello computazionale generalmente sincrono
- Processori vettoriali
  - molte ALU
  - registri vettoriali
  - Unità Load/Store vettoriali
  - Istruzioni vettoriali
  - Memoria interleaved
  - OpenMP, MPI

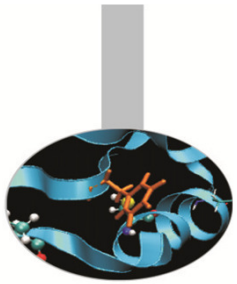
- **G**raphical **P**rocessing **U**nit
  - GPU completamente programmabili
  - molte ALU
  - molte unità Load/Store
  - molte SFU
  - migliaia di threads lavorano in parallelo
  - CUDA

Classificazione architetture - MIMD



- Molteplici streams di istruzioni eseguiti simultaneamente su molteplici streams di dati
  - modello computazionale asincrono
- Cluster
  - molti nodi di calcolo (centinaia/migliaia)
  - più processori multicore per nodo
  - RAM condivisa sul nodo
  - RAM distribuita fra i nodi
  - livelli di memoria gerarchici
  - OpenMP, MPI, MPI+OpenMP

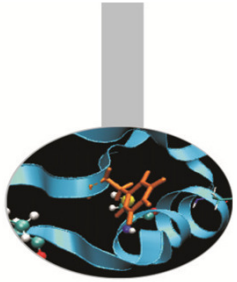
Soluzioni MIMD - IBM - Blue Gene/Q



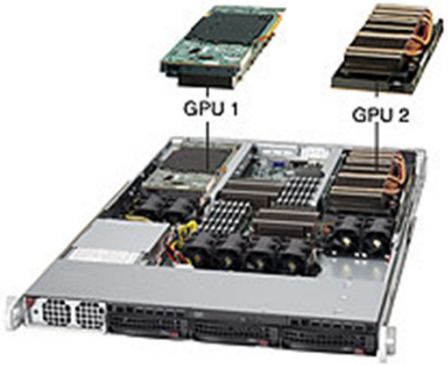
Fermi : 15° posto nella TOP 500

Model: IBM-BlueGene /Q
Architecture: 10 BGQ Frame with 2 MidPlanes each
Front-end Nodes OS: Red-Hat EL 6.2
Compute Node Kernel: lightweight Linux-like kernel
Processor Type: IBM PowerA2, 16 cores, 1.6 GHz
Computing Nodes: 10.240
Computing Cores: 163.840
RAM: 16GB / node
Internal Network: Network interface
with 11 links ->5D Torus
Disk Space: more than 2PB of scratch space
Peak Performance: 2.1 PFlop/s

Soluzioni Ibride - Cluster CPU-GPU

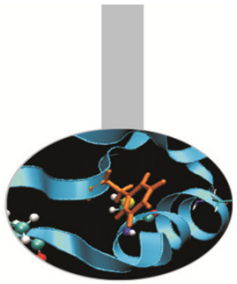


- Soluzioni ibride CPU multi-core + GPU many-core:
  - ogni nodo di calcolo è dotato di processori multicore e schede grafiche con processori dedicati per il GPU computing
  - notevole potenza di calcolo teorica sul singolo nodo
  - ulteriore strato di memoria dato dalla memoria delle GPU
  - OpenMP, MPI, CUDA e soluzioni ibride MPI+OpenMP, MPI+CUDA, OpenMP+CUDA, OpenMP+MPI+CUDA

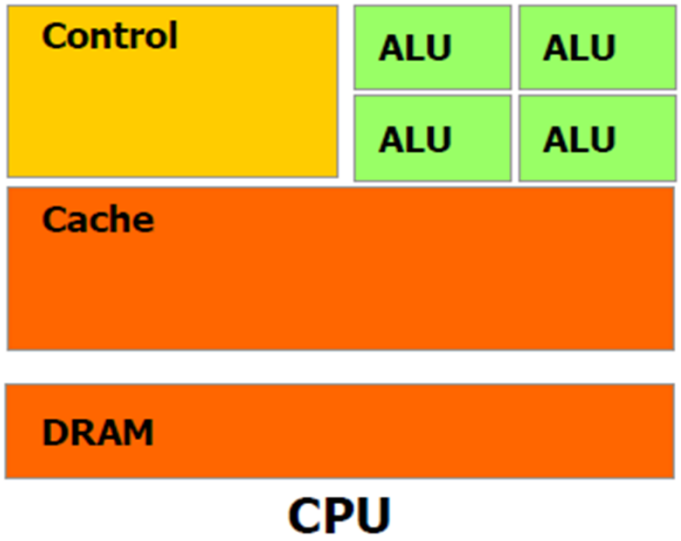


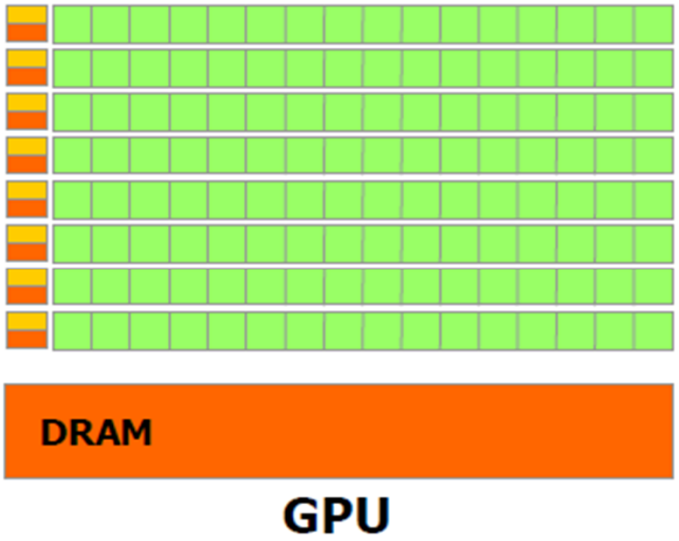
CINECA - PLX

CPU vs GPU

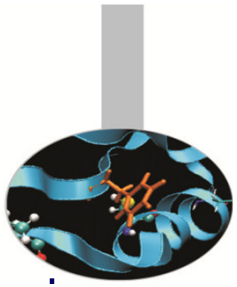


- Le CPU sono processori general purpose in grado di risolvere qualsiasi algoritmo
  - threads in grado di gestire qualsiasi operazione ma pesanti, al massimo 1 thread per core computazionale.
- Le GPU sono processori specializzati per problemi che possono essere classificati come «intense data-parallel computations»
  - controllo di flusso molto semplice (control unit ridotta)
  - molti threads leggeri che lavorano in parallelo

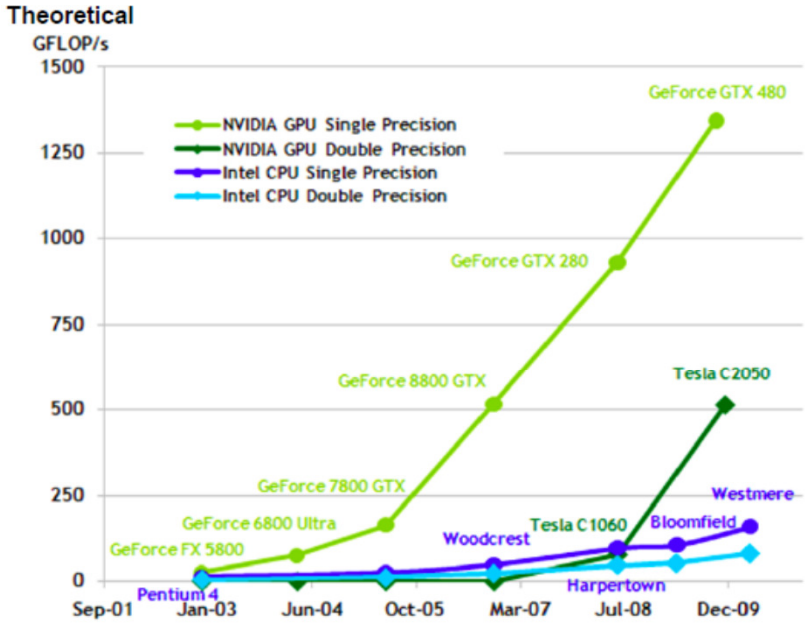




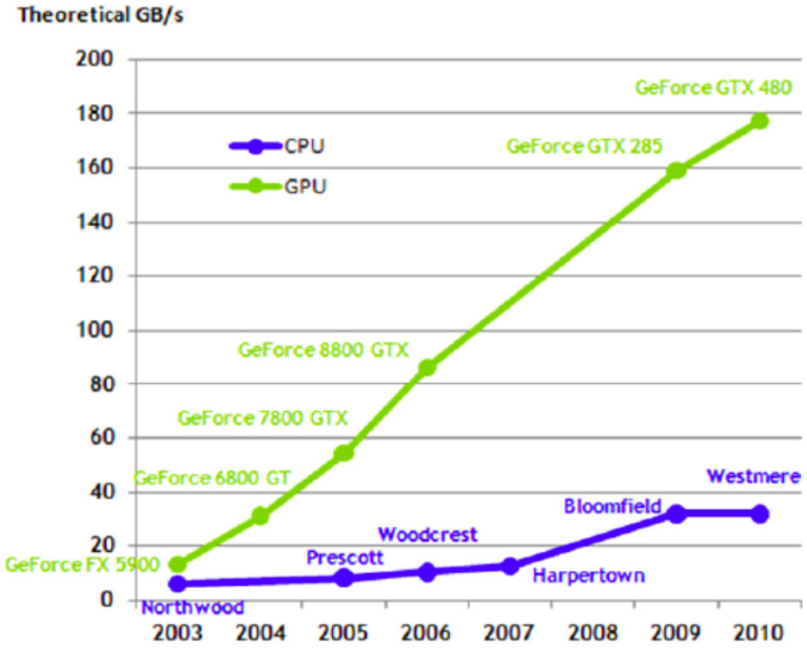
Multicore vs Manicore



- Una nuova direzione di sviluppo per l'architettura dei microprocessori:
  - incrementare la potenza di calcolo complessiva tramite l'aumento del numero di unità di elaborazione piuttosto che della loro potenza
  - la potenza di calcolo e la larghezza di banda delle GPU ha sorpassato quella delle CPU di un fattore 10.

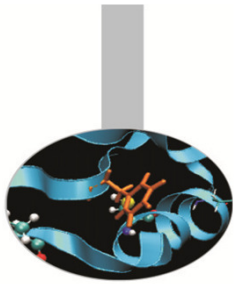


Numero di operazioni in virgola mobile al secondo per la CPU e la GPU



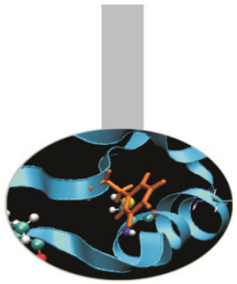
Larghezza di banda della memoria

Coprocessore Intel Xeon Phy

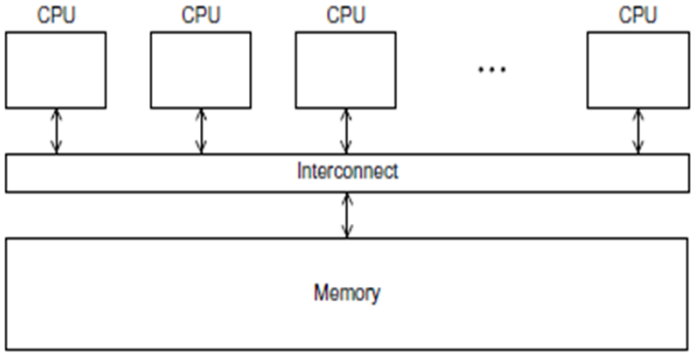


- **Specifiche principali:**
  - Basato su architettura Intel Many Integrated Core (MIC)
  - 60 core / 1,053 GHz / 240 thread
  - 8 GB di memoria e larghezza di banda di 320 GB/s
  - 1 TFLOPS di prestazioni di picco a doppia precisione
  - Fattore di forma PCIe x16 standard, raffreddamento passivo
  - Sistema operativo Linux*, IP indirizzabile
  - Istruzioni SIMD a 512 bit
  - Approcci tradizionali come MPI, OpenMP

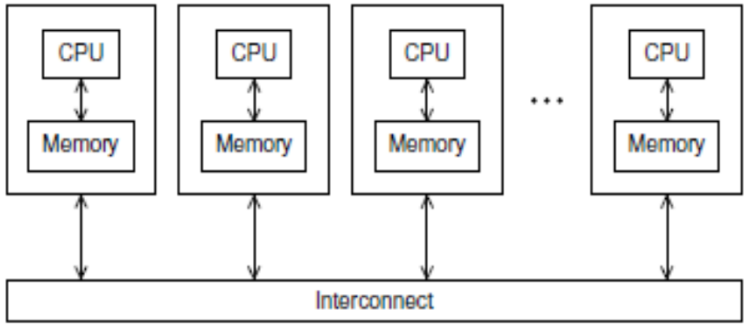
Memoria Condivisa - Memoria Distribuita



- Le architetture MIMD classiche e quelle miste CPU – GPU sono suddivise in due categorie
  - Sistemi a **memoria condivisa** dove ogni singolo core ha accesso a tutta la memoria
  - Sistemi a **memoria distribuita** dove ogni processore ha la sua memoria privata e comunica con gli altri tramite scambio di messaggi. I moderni sistemi multicore hanno memoria condivisa sul nodo e distribuita fra i nodi.

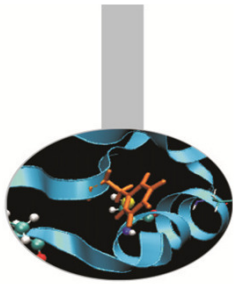


Memoria condivisa

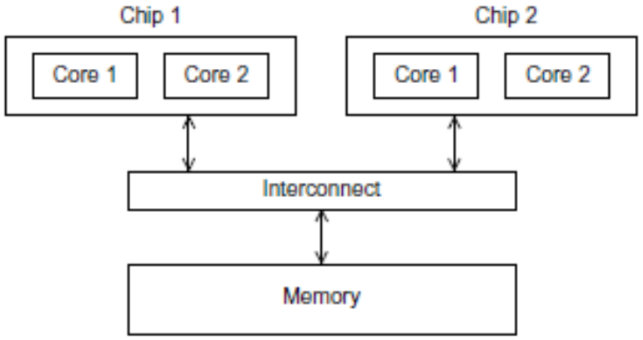


Memoria distribuita

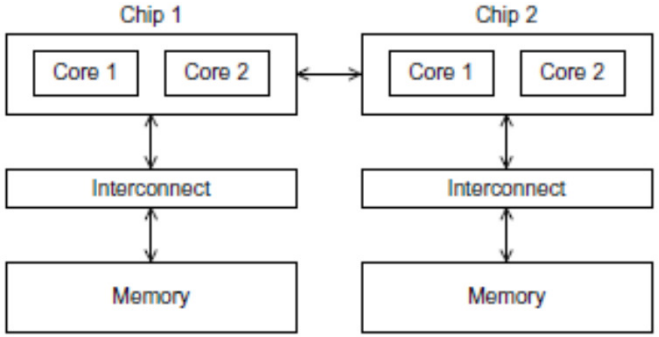
Memoria Condivisa



- Nelle architetture a memoria condivisa con processori multicore vi sono generalmente due tipologie di accesso alla memoria principale
  - **U**niform **M**emory **A**ccess dove tutti i cores sono direttamente collegati con la stessa priorità alla memoria principale tramite il sistema di interconnessione
  - **N**on **U**niform **M**emory **A**ccess dove ogni processore multicore può avere accesso privilegiato ad un blocco di memoria e accesso secondario agli altri blocchi.

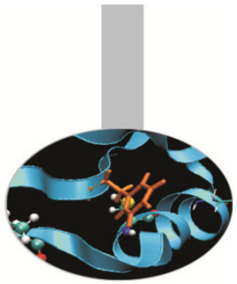


Uniform Memory Access

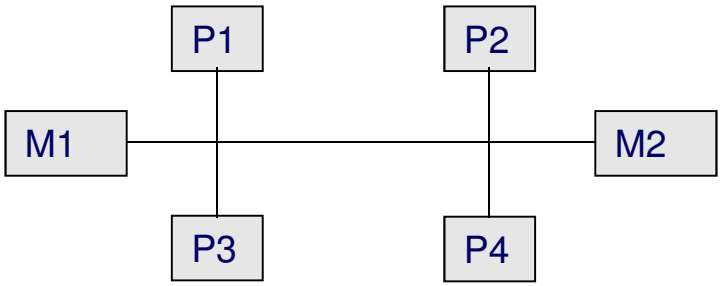


Non Uniform Memory Access

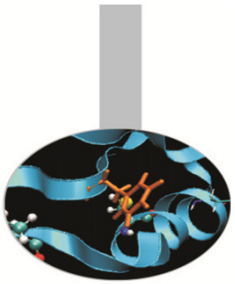
Reti di interconnessione Sistemi memoria condivisa



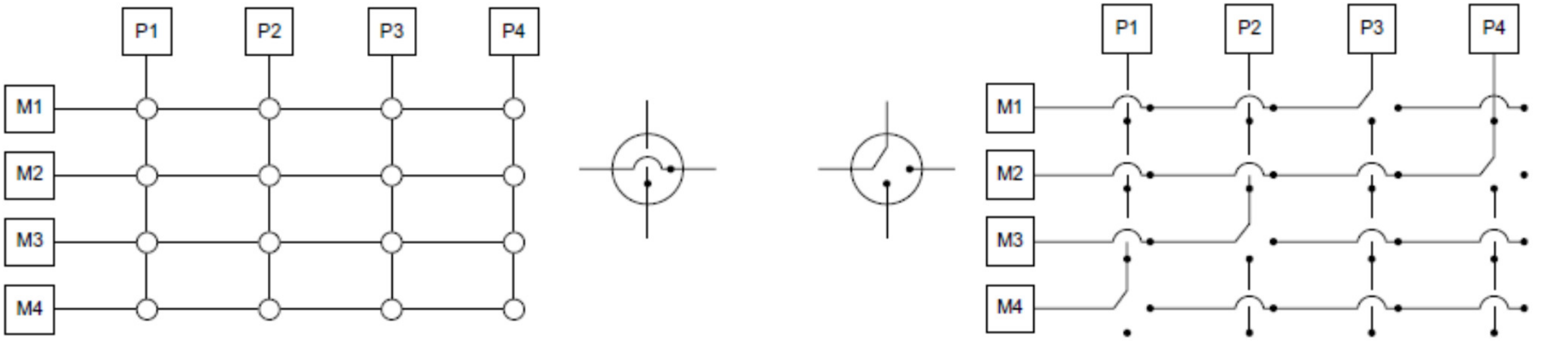
- Nei sistemi a memoria condivisa le due soluzioni maggiormente utilizzate sono **buses** e **crossbars**.
- Il **bus** è costituito da una serie di canali di comunicazione paralleli corredati da un sistema di controllo di accesso. I canali sono condivisi da tutti i device che sono collegati al bus
  - basso costo
  - ridotto numero di device collegabili
  - possibli conflitti di accesso direttamente proporzionali al numero di processori collegati
  - usati usualmente per la comunicazione cores - memoria locale ai cores



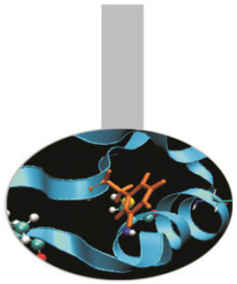
Reti di interconnessione Sistemi memoria condivisa



- Il sistema **crossbar switch** è la metodologia più frequente che si usa per collegare diversi device in shared memory
  - le linee di comunicazione sono link bidirezionali
  - gli switches possono assumere due configurazioni
  - se il rapporto banchi memoria/processori è 1/1 allora ci sarà conflitto solo se 2 cores cercano di accedere allo stesso modulo di memoria contemporaneamente

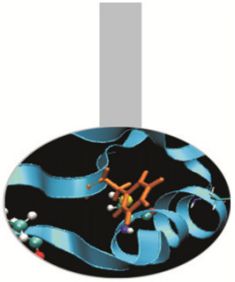


Reti di interconnessione Sistemi memoria distribuita

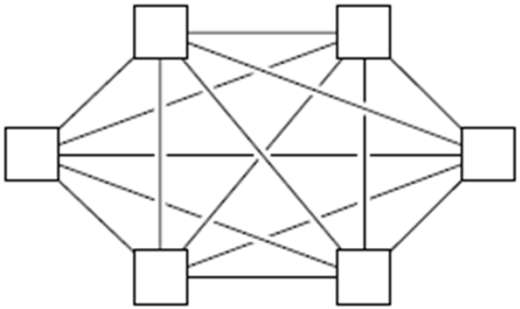


- Tutti i sistemi caratterizzati da elevate potenze di calcolo sono composti da diversi nodi, a memoria condivisa sul singolo nodo e distribuita fra i nodi
  - i nodi sono collegati fra di loro da topologie di interconnessione più o meno complesse e costose
- Le reti di interconnessione commerciali maggiormente utilizzate sono
  - Gigabit Ethernet : la più diffusa, basso costo, basse prestazioni
  - Infiniband : molto diffusa, elevate prestazioni, costo elevato (50% del costo di un cluster)
  - Myrinet : sempre meno diffusa dopo l'avvento di infiniband, vi sono comunque ancora sistemi HPC molto importanti che la utilizzano
- Reti di interconnessione di nicchia
  - Quadrics
  - Cray

Reti di interconnessione Sistemi memoria distribuita

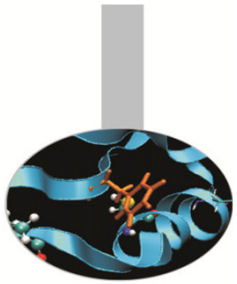


- La topologia ideale è **l'interconnessione completa** dove ogni switch è direttamente collegato a tutti gli altri, ovvero ogni nodo ha una connessione diretta con tutti gli altri.
  - topologia troppo costosa e impossibile da progettare per sistemi costituiti da molti nodi, viene usata come termine di paragone per verificare l'efficienza delle topologie reali
  - soluzioni reali sono: **anello**, **toroide**, **ipercubo**, **omega**, **fat tree**.
  - la soluzione meno costosa e meno efficiente è la **LAN**.



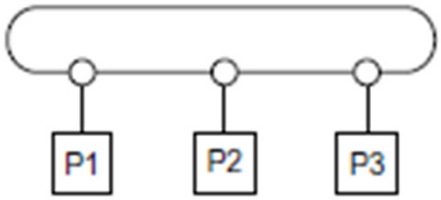
Interconnessione completa

Anello - Toroide

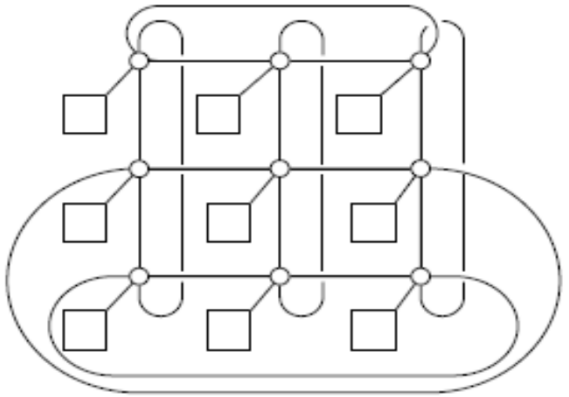


- L'interconnessione ad anello è superiore alla semplice LAN perché permette più connessioni simultanee
  - dati **p** processori si hanno **2p** links.

- L'interconnessione toroidale può essere bidimensionale, tridimensionale o pentadimemsionale (Fermi)
  - più performante della connessione ad anello perché aumenta il numero di connessioni .

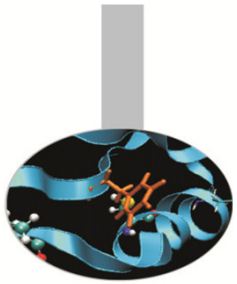


Anello

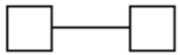


Toroide bidimensionale

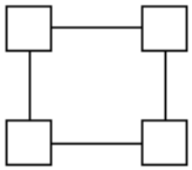
Ipercubo



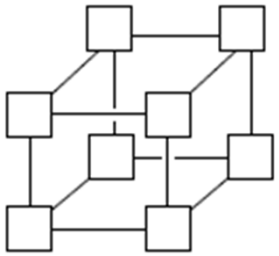
- L'interconnessione ad **ipercubo** è una topologia molto efficiente che prevede una buona connettività ed è molto utilizzata nei più importanti sistemi HPC
  - l'ipercubo monodimensionale è un sistema completamente connesso costituito da due processori
  - l'ipercubo bidimensionale è costruito partendo da due ipercubi monodimensionali collegando gli switch corrispondenti
  - l'ipercubo tridimensionale è costituito da bue ipercubi bidimensionali
  - un ipercubo di dimensione **d** ha p=2^d nodi



Ipercubo monodimensionale

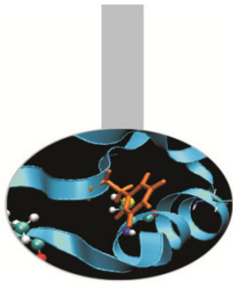


Ipercubo bidimensionale

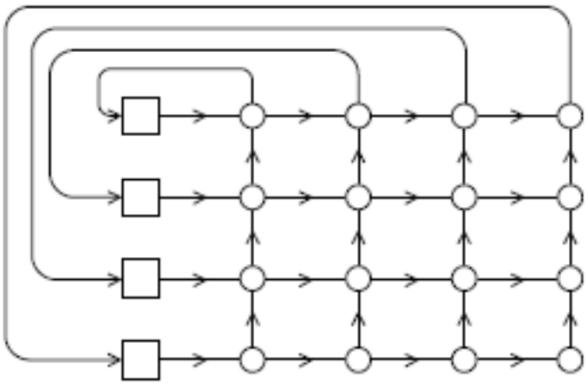


Ipercubo tridimensionale

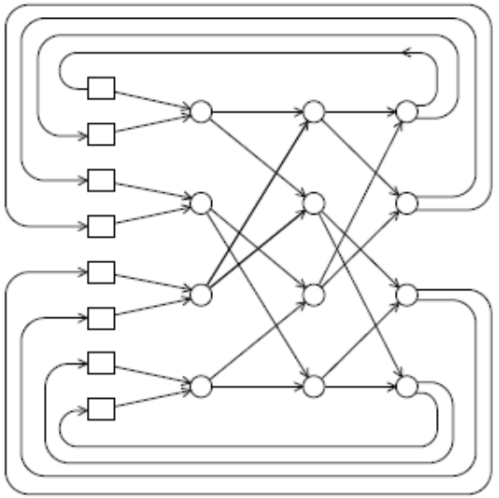
Omega - Crossbar



- E' una topologia che deriva dalla topologia crossbar switch 2 x 2 ma ha una connettività minore
  - costo inferiore rispetto alla crossbar
  - la comunicazione fra alcune coppie di nodi disabilita la possibiltà ad altre coppie di comunicare contemporaneamente.
  - dati **p** nodi il numero di switch utilizzato è $2p \bullet \log_2 p$
  - nella rete crossbar il numero di switch è pari a p^2

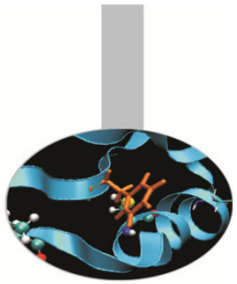


Crossbar 2 x 2

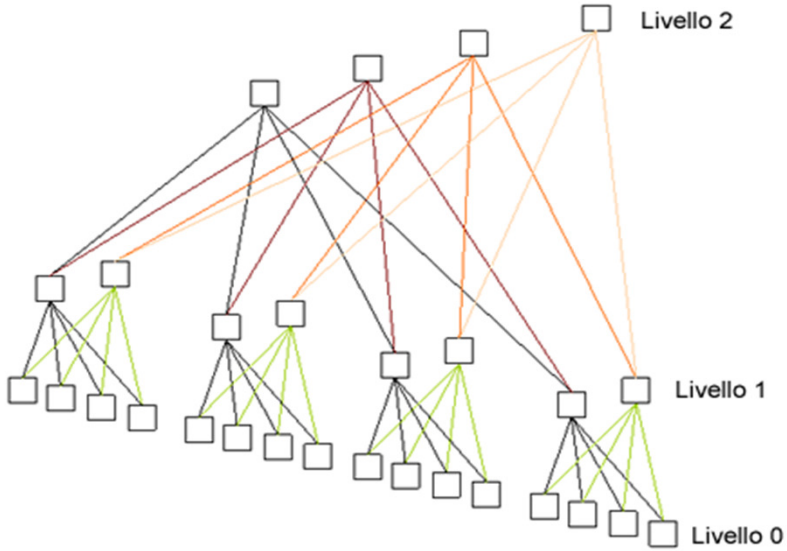


Omega

Fat tree

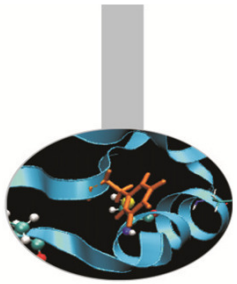


- Un' interconnessione ***fat tree*** è una rete in cui il numero di livelli è dato dalla formula: $\log_4 p$ dove *p* indica il numero di nodi. Ogni nodo possiede 4 connessioni con il livello inferiore e 2 con quello superiore.



Interconnessione Fat tree

Top 500



Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
15	CINECA Italy	**Fermi** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	163840	1788.9	2097.2	822