Architetture parallele

Architetture Avanzate dei Calcolatori

Valeria Cardellini

Oltre l'ILP

- Le architetture multiple-issue supportano un parallelismo *a grana fine*, a livello di istruzione
- Incrementare le prestazioni del singolo processore è sempre più difficile
  - Diminuiscono i vantaggi dell'ILP (difficile estrarre ulteriore parallelismo)
  - Problema del consumo di potenza
  - Tendenza: passare a molti processori (più semplici) su un singolo chip
- Un passo più in là:
  - Multithreading
  - Architetture parallele
    - Si collegano più processori in un sistema complesso

Multithreading

- Il processore esegue contemporaneamente più thread
  - Ogni thread ha le sue istruzioni ed i suoi dati
  - Molteplici thread condividono le unità funzionali di un singolo processore sovrapponendosi nell'esecuzione
  - Il processore deve mantenere una copia distinta dello stato di ciascun thread (registri, PC, tabella delle pagine)
- Multithreading a grana fine
  - Il processore esegue un'istruzione per ogni thread
  - Svantaggio: rallenta l'esecuzione del singolo thread
- Multithreading a grana grossa
  - Il processore cerca di eseguire più istruzioni per thread, passando ad un altro thread in occasioni di stalli lunghi
- Multithreading simultaneo
  - Raffinamento del multithreading a grana grossa
  - Anche noto come hyperthreading

Architetture parallele

- Definizione (Almasi e Gottlieb, 1989): un'architettura parallela è un insieme di elementi di elaborazione che *cooperano* e *comunicano* per risolvere *velocemente* problemi di dimensioni considerevoli, talvolta intrattabili su macchine sequenziali

- Obiettivi:
  - Migliorare le prestazioni
    - E' possibile utilizzare il calcolo parallelo per risolvere
      - Un problema più grande nello stesso tempo (***scale-up***)
      - Lo stesso problema in minor tempo (***speed-up***)
  - Migliorare il rapporto costo/prestazioni

Architetture parallele (2)

- Esempi di applicazioni
  - Calcolo scientifico
    - Previsioni meteorologiche
    - Simulazione realistica di veicoli (crash testing)
  - Motori Web di ricerca
  - Entertainment/grafica
- Il calcolo parallelo è l'unico strumento in grado di affrontare le grandi simulazioni del mondo fisico

- Opzioni di progettazione di un'architettura parallela:
  - Quanti processori?
  - Quale è la capacità di elaborazione di ciascun processore?
  - Quale tecnologia utilizzano i processori?
  - Quale organizzazione per l'elaborazione?
  - Come è organizzata la memoria?
  - In che modo sono interconnessi i processori?
  - Come vengono scambiate le informazioni?

Le sfide del calcolo parallelo

- Il calcolo parallelo fino ad ora si è sviluppato soprattutto in ambito accademico ed in centri di ricerca: perché?
- Primo problema: parallelismo limitato nei programmi
- Secondo problema: elevata latenza degli accessi remoti
- La vera difficoltà non è realizzare architetture parallele, ma sviluppare applicazioni parallele
  - Le architetture parallele raggiungono buone prestazioni solo se programmate in modo opportuno
  - Servono nuovi algoritmi
  - Gli ambienti di programmazione non sono molto sviluppati
    - La programmazione parallela è più complessa
    - Il debugging di programmi paralleli è complesso
    - I super-compilatori autoparallelizzanti sono solo sperimentali

Tassonomia tradizionale

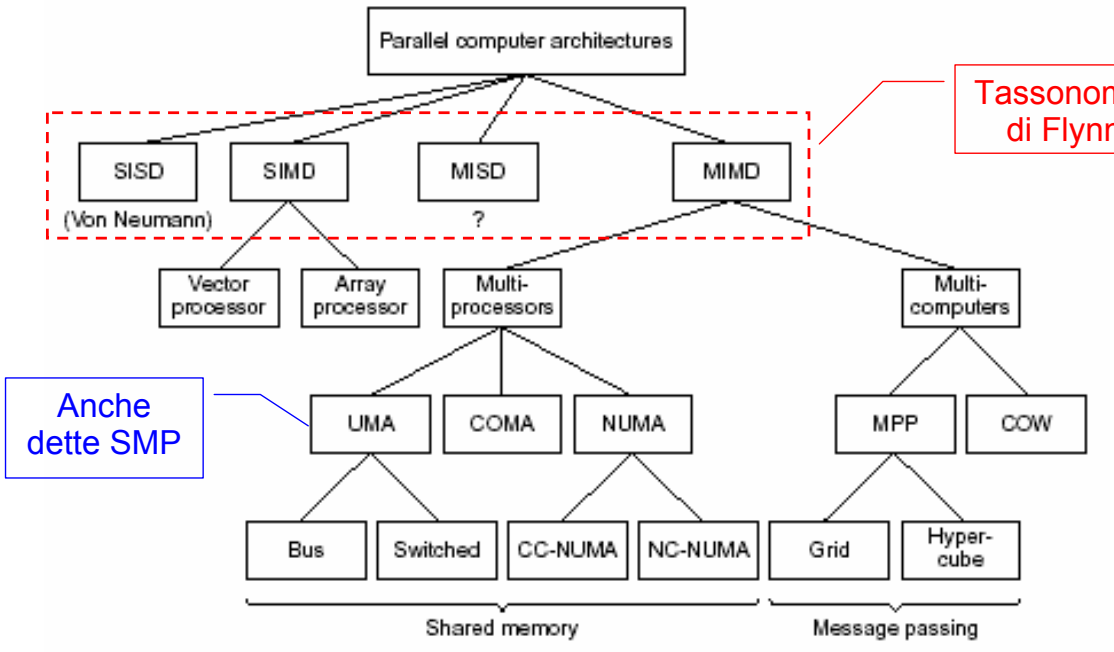
- Ad opera di Flynn (1966)
- Single instruction stream, single data stream (SISD)
  - Il processore singolo convenzionale (architettura di Von Neumann)
- Single instruction stream, multiple data stream (SIMD)
  - Una stessa istruzione eseguita da più processori che usano diversi flussi di dati: *parallelismo a livello di dati*
  - Ogni processore ha la propria memoria dati; ci sono una sola memoria istruzioni ed un solo processore di controllo
  - I processori multimediali presentano una forma (limitata) di parallelismo SIMD; le architetture vettoriali sono la classe più ampia
- Multiple instruction stream, single data stream (MISD)
  - Non è mai stato costruito nessun multiprocessore commerciale di questo tipo

Tassonomia tradizionale (2)

- Multiple instruction stream, multiple data stream (MIMD)
  - Ogni processore legge le proprie istruzioni ed opera sui propri dati: *parallelismo a livello di thread*
  - I processori sono spesso commerciali standard (*commodity*)
- Il modello MIMD è emerso come l'architettura prescelta di tipo generale perché:
  - è flessibile
  - si possono costruire architetture MIMD usando processori commerciali

Tassonomia estesa

Estensione della tassonomia di Flynn per considerare diversi tipi di architetture MIMD



Architetture MIMD

- **Multiprocessori**
  - Architetture MIMD a **memoria condivisa** (shared memory): memoria con un unico spazio di indirizzamento, condiviso tra tutti i processori
  - I processori comunicano tramite *variabili condivise* e ciascun processore è in grado di accedere ad ogni locazione di memoria tramite operazioni di *load* e *store*
  - Occorre un meccanismo di *sincronizzazione*, per coordinare il comportamento di processi che possono essere in esecuzione su processori differenti
- **Multicomputer**
  - Architetture MIMD a **memoria distribuita** (distributed memory): memorie riservate dei singoli processori con uno spazio di indirizzamento costituito da più spazi privati, logicamente disgiunti e non indirizzabili da parte di un processore remoto
  - La comunicazione tra i processori avviene mediante *scambio di messaggi* (message passing) esplicito
    - Procedure per inviare e ricevere messaggi (send e receive)

Classificazione dei multiprocessori

- Architetture **UMA** (Uniform Memory Access)
  - Anche dette **SMP** (Symmetric MultiProcessor): multiprocessori simmetrici o multiprocessori con memoria ad accesso uniforme
  - Ogni processore ha lo stesso tempo di accesso alla memoria principale, indipendentemente dal processore richiedente e dalla locazione della parola di memoria richiesta
  - La memoria ha una relazione simmetrica con tutti i processori
- Architetture **NUMA** (Non Uniform Memory Access)
  - Multiprocessori con memoria ad accesso non uniforme
  - Alcuni accessi in memoria sono più veloci di altri a seconda del processore richiedente e dalla locazione della parola di memoria richiesta
- Architetture **COMA** (Cache Only Memory Access)
  - L'accesso avviene solo tramite cache

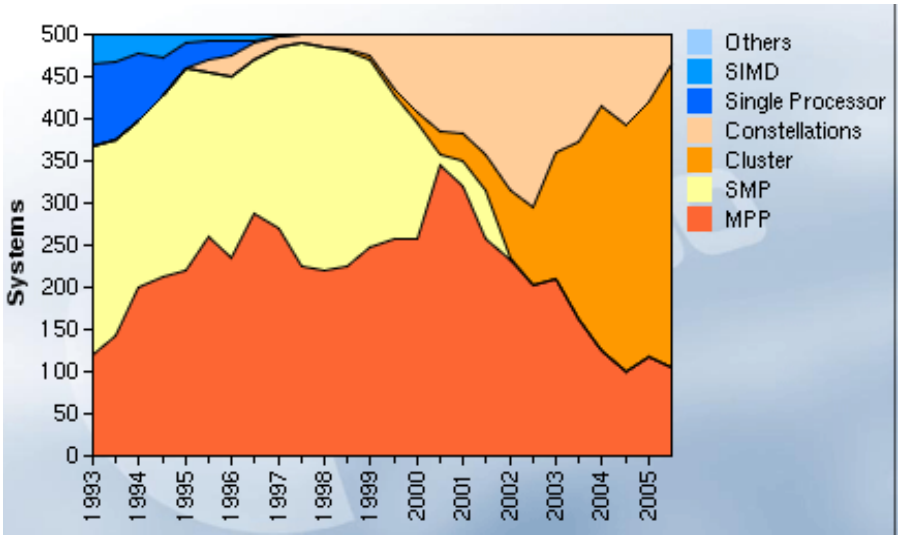
Classificazione dei multicomputer

- Architetture **MPP** (Massively Parallel Processor)
  - Nodi autonomi connessi con reti proprietarie, veloci, a banda larga e bassa latenza
- **Cluster**
  - Computer totalmente separati, collegati tramite una rete locale ad alta banda (tipicamente tecnologia *off-the-shelf*)
  - Approccio molto efficiente in termini di costo
  - Anche indicati con i termini
    - NOW (Network of Workstations)
    - COW (Cluster of Workstations)

Diffusione delle varie architetture nei top 500

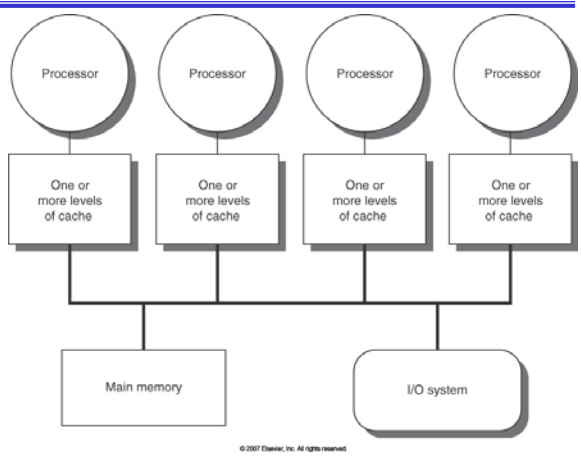
- Lista dei 500 top supercomputer:
  - http://www.top500.org
  - Prestazioni misurate tramite il benchmark Linpack (soluzione di un sistema denso di equazioni lineari Ax=b)
  - Classifica aggiornata ogni 6 mesi

In un decennio:
- scomparsa dai top 500 di uniprocessori, SMP e SIMD
- crescita dei cluster



Architettura SMP di base

- Tutti i processori hanno un tempo di accesso uniforme alla memoria
  - La **memoria condivisa** è suddivisa fisicamente in banchi
  - La memoria ed i processori sono collegati tramite uno (o più) bus

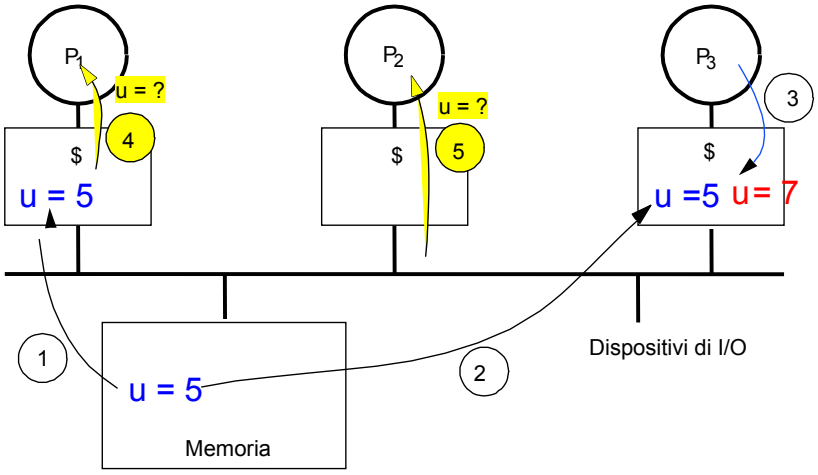


- La comunicazione tra processori avviene mediante un ampio spazio di indirizzamento condiviso
  - Occorre gestire i conflitti per l'accesso alla memoria da parte di più processori
  - Occorre sincronizzare gli accessi alla stessa locazione di memoria da parte di più processori
- Dimensione tipica: da 2 a 32 processori

Coerenza della cache nell'architettura SMP

- La cache locale per ciascun processore permette di ridurre il traffico sul bus e la latenza di accesso ai dati
  - Bus singolo: collo di bottiglia dell'architettura SMP
  - I riferimenti in memoria che si concludono con un cache hit non richiedono un accesso alla memoria condivisa
- La cache locale può contenere:
  - dati *privati*: usati solo sullo stesso processore
  - dati *condivisi*: usati da più processori e che sostanzialmente forniscono il meccanismo di comunicazione tra i processori
- Accesso ad un dato condiviso: il dato può essere replicato in più cache
  - Nascono problemi di **coerenza della cache**!
  - Due processori diversi vedono la memoria tramite la propria cache ma le due viste possono essere diverse

Problema della coerenza della cache: esempio



- I processori vedono valori diversi di u dopo l'evento 3
- Con cache di tipo write back, non è predicibile quando il nuovo valore di u verrà scritto in memoria
  - I processori che accedono in memoria possono vedere un vecchio valore
  - Inaccettabile per la programmazione!

Coerenza e consistenza

- Informalmente: un sistema di memoria è coerente se qualsiasi lettura di un dato fornisce il valore del dato scritto più recentemente
- Coerenza: definisce quali valori devono essere restituiti nel caso di lettura
- Consistenza: determina quando un valore che è stato scritto verrà restituito da una lettura

Coerenza del sistema di memoria

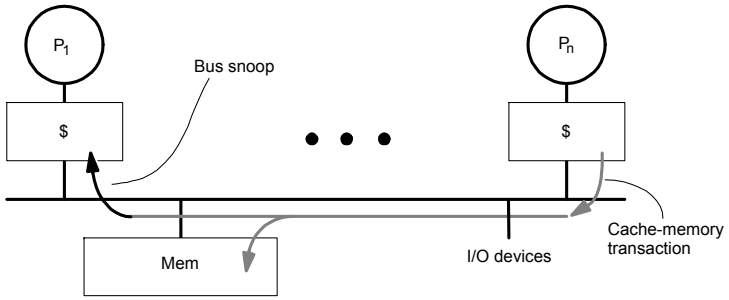
- Il sistema di memoria è coerente se:
  1. Preserva l'ordine di programma: una lettura da parte di un processore P in una locazione X che segue una scrittura in X da parte di P (senza scritture intermedie in X da parte di un altro processore) restituisce sempre il valore scritto da P
  2. Vista coerente della memoria: una lettura da parte di un processore in una locazione X che segue una scrittura in X da parte di un altro processore restituisce il valore scritto se lettura e scrittura sono separate da un tempo sufficiente e se non si verificano altre scritture in X tra i due accessi
  3. Serializzazione delle scritture: le scritture di una stessa locazione X sono *serializzate*, ovvero due scritture in X da parte di una qualsiasi coppia di processori sono viste da tutti gli altri processori nello stesso ordine

Soluzioni hardware per la coerenza

- Le soluzioni hardware per garantire la coerenza della cache sono dette *protocolli di coerenza della cache*
  - Riconoscimento dinamico (a tempo di esecuzione) delle potenziali condizioni di incoerenza
  - Alternativa alle soluzioni hardware: soluzioni software (intervento del compilatore e del sistema operativo)
- Due categorie di schemi hardware:
  - **Protocolli di directory**
    - Directory centralizzata che contiene informazioni globali sullo stato di condivisione dei blocchi di memoria presenti nelle cache locali
    - Controllore centralizzato responsabile della coerenza delle cache
  - **Protocolli di snooping**
    - Ogni cache con una copia dei dati possiede anche una copia dello stato di condivisione del blocco
    - La responsabilità della coerenza della cache è distribuita tra i controllori di cache

Protocollo di snooping

- E' la classe di protocolli più popolare per garantire la coerenza della cache
- Ogni cache che ha una copia di un dato letto da un blocco di memoria fisica ha anche una copia del suo stato di condivisione
- Tutte le cache sono collegate alla memoria condivisa tramite un bus
- Tutti i controllori delle cache monitorano o "sbirciano" (*snoop*) il bus, per verificare se hanno una copia del blocco di cache richiesto sul bus



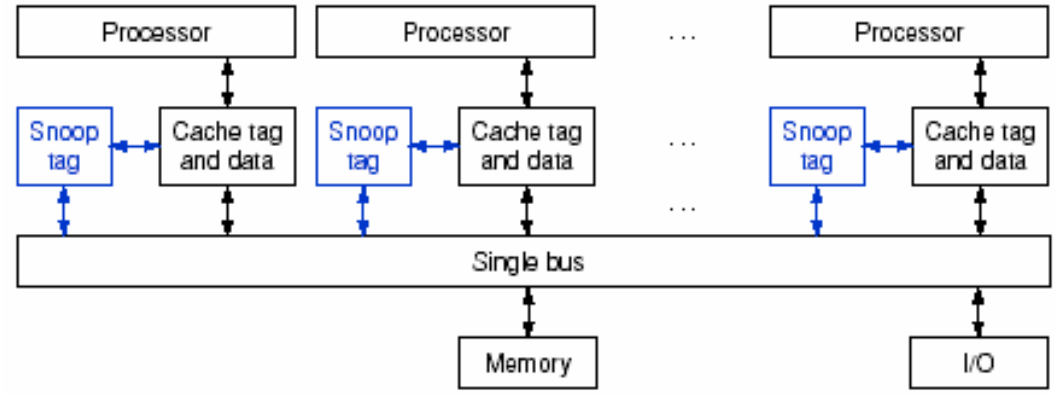
Protocollo di snooping (2)

- In seguito ad una scrittura su un dato condiviso occorre scegliere se adottare una strategia di:
  - **invalidazione** oppure **aggiornamento** delle copie
- Invalidazione (*write-invalidate protocol*)
  - tutte le copie presenti nelle altre cache sono invalidate
- Aggiornamento (*write-update protocol*)
  - tutte le copie presenti nelle altre cache sono aggiornate

- Il protocollo write-update consuma più banda per l'aggiornamento: tutti i multiprocessori più recenti usano un protocollo di tipo write-invalidate

Protocollo di snooping (3)

- Per implementare il protocollo di snooping vengono ampliati i bit di gestione dello stato già presenti in ogni blocco di cache (valid, dirty)

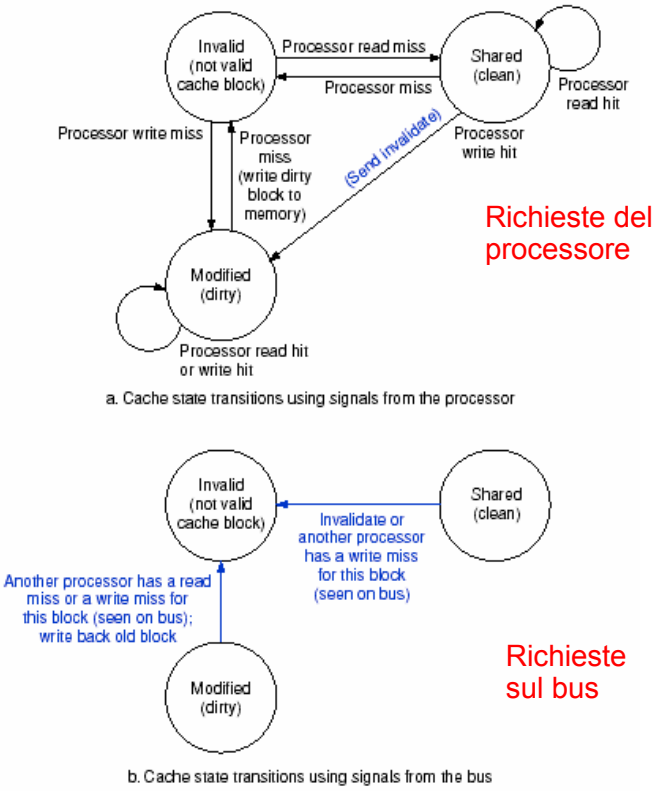


Individuare la copia aggiornata di un dato

- Strategia di scrittura write-through: si ottiene il dato aggiornato dalla memoria
  - Più semplice se vi è una banda di memoria sufficiente
- Strategia di scrittura write-back: la copia più recente del dato può essere in una cache
  - Più difficile
- Possono usare lo stesso meccanismo di snooping
  1. Controllare ogni indirizzo in transito sul bus
  2. Se un processore possiede una copia dirty del blocco di cache richiesto, lo fornisce in risposta ad una richiesta di lettura ed interrompe l'accesso in memoria
  - Più complesso ottenere il blocco da una cache; può essere necessario più tempo rispetto ad ottenerlo dalla memoria
- Write-back richiede meno banda di memoria
  - Supporta un maggior numero di processori
  - E' la strategia usata dalla maggior parte dei multiprocessori

Protocollo di snooping con write-invalidate

- Esempio di protocollo con write-invalidate basato su una strategia di scrittura di tipo write-back
- Ciascun blocco di cache è in uno dei 3 stati:
  - **shared** (solo letto): blocco pulito e può essere condiviso
  - **modified** (letto/scritto): blocco sporco e *non* può essere condiviso
  - **invalid**: blocco non contenente dati validi
- Diagramma di transizione a stati finiti per ogni blocco di cache



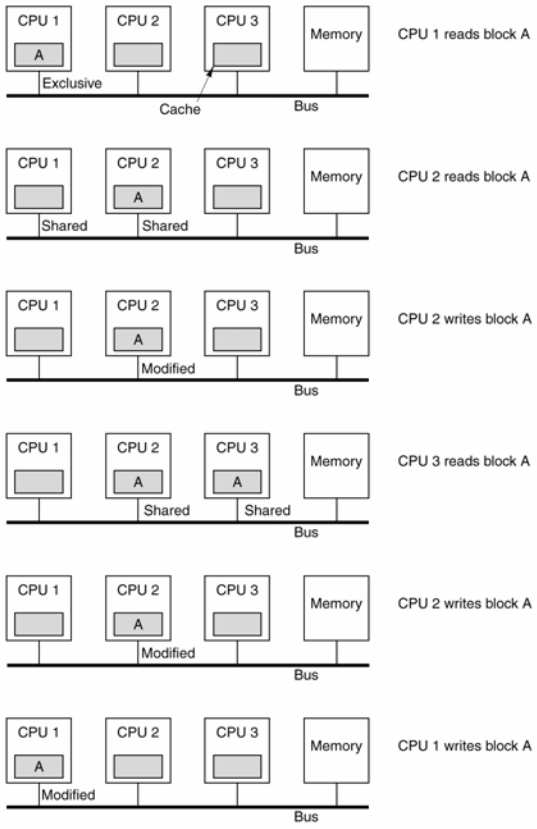
Protocollo MESI

- Esempio di protocollo di coerenza della cache con write-back: i blocchi sono aggiornati in memoria solo quando un processore finisce di scriverci
  - Usato da Pentium 4 e PowerPC
- Stati di un blocco di cache:
  - **invalid** (non valido): blocco in cache non contiene dati validi
  - **shared** (condiviso): blocco presente in più cache, memoria aggiornata
  - **exclusive** (esclusivo): blocco presente solo in quella cache, memoria aggiornata
  - **modified** (modificato): blocco presente solo in quella cache, memoria non aggiornata
- Il nome del protocollo è l'acronimo dei 4 stati

Protocollo MESI (2)

• La lettura di un blocco modificato obbliga il richiedente ad attendere ed il possessore ad aggiornare la memoria (il blocco ridiventa condiviso)

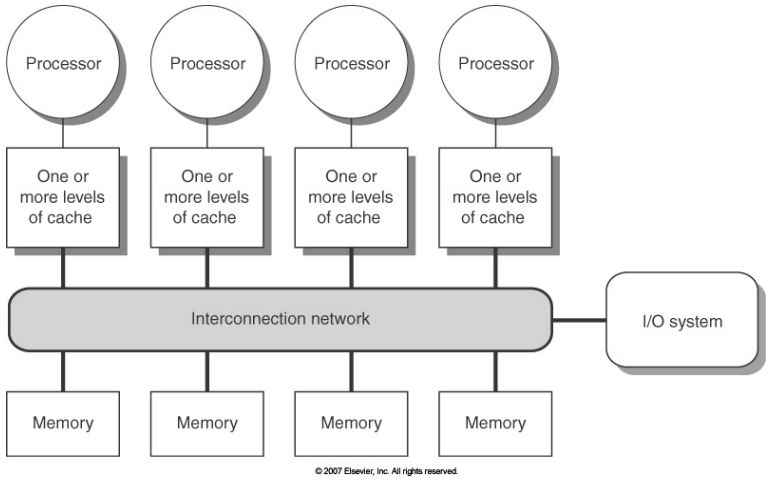
• La scrittura di un blocco modificato obbliga il possessore a marcare il blocco come non valido dopo averlo copiato in memoria



Architetture multi-core

- Variante dell'architettura multi-processore, in cui si hanno più core in un solo chip
  - Core (nucleo, cuore): processore, la rispettiva cache ed il cache controller
- Il socket della schema madre (l'alloggiamento del processore sulla schema madre) è lo stesso ma sul package vengono montati più core identici
  - 2 core nel caso del dual-core
  - nel 2007 8 core

Architettura a memoria distribuita



- Rete di interconnessione: soluzione indispensabile per gestire un numero elevato di processori
  - E' una struttura più scalabile e complessa del bus
  - Anche per multiprocessori, non solo per multicomputer

Architettura a memoria distribuita (2)

- Due approcci architetturali alternativi
- 1° approccio: la comunicazione avviene ancora mediante uno spazio di indirizzamento condiviso
  - Memorie fisicamente separate ma indirizzate come un unico spazio condiviso
  - Qualunque processore può accedere a qualunque locazione di memoria
  - Dette architetture a *memoria distribuita condivisa* (Distributed Shared Memory o DSM) o anche **NUMA** (Non Uniform Memory Access)
- 2° approccio: la comunicazione avviene tramite scambio di messaggi
  - Lo spazio di indirizzamento consiste di più spazi di indirizzamento privati, logicamente disgiunti e che non possono essere indirizzati da un processore remoto
  - Dette architetture multicomputer

Reti di interconnessione

- Le reti di interconnessione possono essere statiche o dinamiche
  - Rete *statica*: collegamento fisso di unità di commutazione
    - Detta anche rete diretta
  - Rete *dinamica*: unità di commutazione attive possono essere posizionate per riconfigurare le connessioni
- In genere:
  - Memoria condivisa: reti dinamiche
  - Memoria distribuita: reti statiche

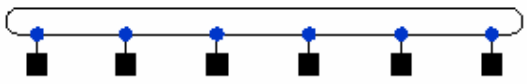
Topologia di interconnessione

- La topologia di una rete di interconnessione, tipicamente modellata come un grafo regolare, definisce i canali (archi del grafo) mediante cui sono collegati i nodi di elaborazione (nodi del grafo)

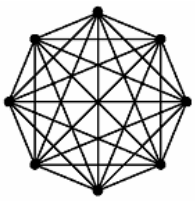
- Principali metriche di prestazione delle topologie
  - Larghezza di banda totale
    - E' la banda di ciascun link (collegamento) per il numero di collegamenti
    - Rappresenta il caso migliore
  - Larghezza di banda di bisezione
    - Si dividono i nodi in due parti, ciascuna delle quali contiene la metà dei nodi
    - E' la somma delle bande dei link che attraversano il taglio
    - Rappresenta il caso peggiore

Esempi di topologie di interconnessione

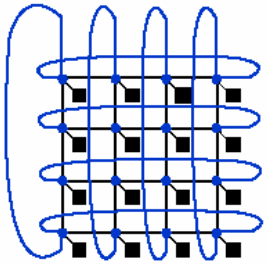
- Consideriamo alcuni esempi di topologie formate da *n* nodi
- Anello
  - Banda totale = *n**banda_link
  - Banda di bisezione = *2**banda_link

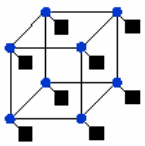


- Completamente connessa
  - Banda totale = (*n**(*n* -1)/2)*banda_link
  - Banda di bisezione = $(n/2)^2$*banda_link



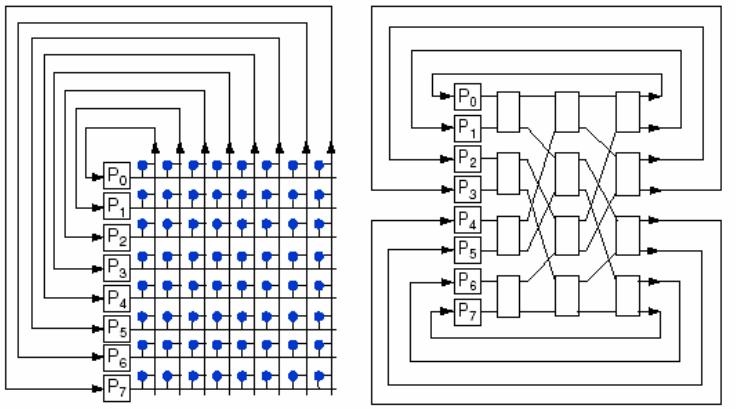
- Griglia 2-D (grid o mesh)
- Cubo 3-D





Esempi di topologie di interconnessione (2)

- Reti multistadio
  - In ciascun nodo uno switch anziché un nodo di elaborazione
- Crossbar e Omega network
  - Crossbar: n^2 switch
  - Omega network: $(n/2)*\log_2 n$ switch (ciascuno switch è formato da 4 switch più piccoli)



a. Crossbar

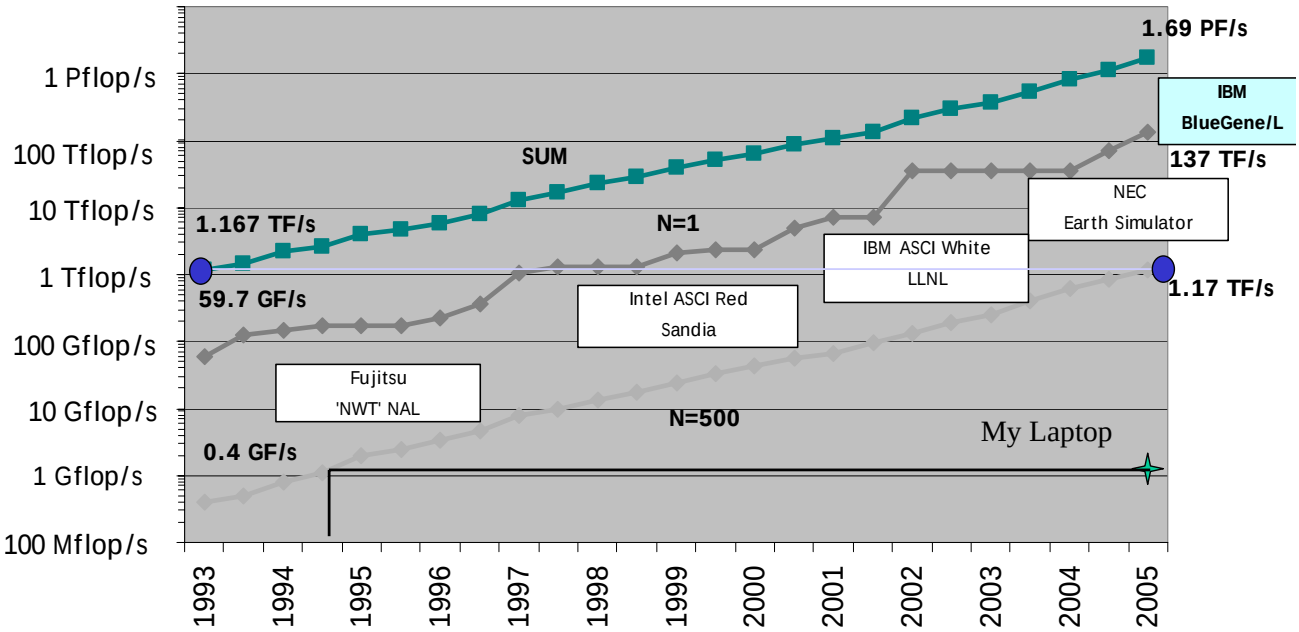
b. Omega network

c. Omega network switch box

La piattaforma di Google

- L'indice di Google comprende oltre 25 miliardi di URL
- Obiettivo: servire una richiesta in meno di 0,5 secondi (ritardi di rete compresi!)
- Centinaia di migliaia di macchine (stima 450000) organizzate in 8 cluster distribuiti geograficamente
  - Nel 2002 Google usava 6000 processori e 12000 dischi, con 2 cluster nella Silicon valley e 2 in Virginia
    - Alcuni petabyte (10^{15}) di spazio su disco
    - Ciascun cluster connesso ad Internet tramite una connessione OC48 (2488 Mbit/sec)
  - Affidabilità (valori del 2002):
    - In un giorno, 20 macchine hanno in media bisogno di riavvio a causa di errori software
    - 2% delle macchine sostituite ogni anno
- I server sono commodity PC con una versione customized di Linux
- Per approfondimenti
  - http://labs.google.com/papers/

TOP500 Performance Trend – June 2005



All systems > 1Tflop, last was at 300 last time; top100 starts at 3.4 tfs; 304 cluster

Courtesy of Jack Dongarra

Architecture/Systems Continuum

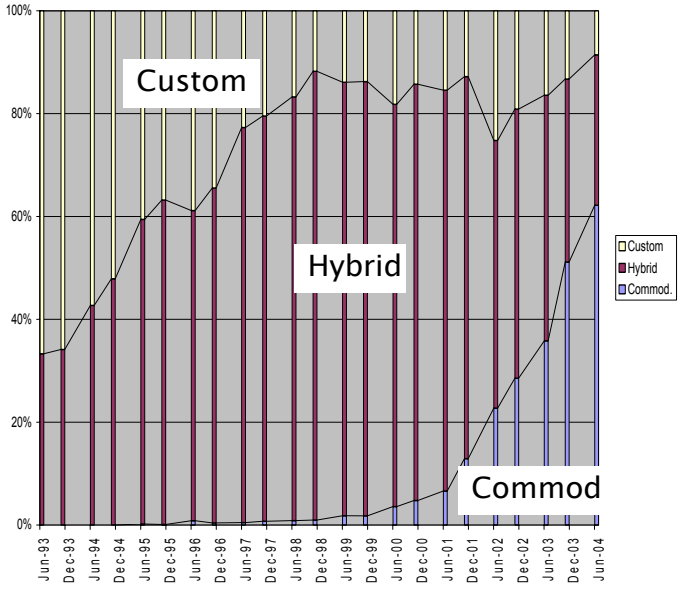
Tightly Coupled

- Custom processor with custom interconnect
  - Cray X1
  - NEC SX-8
  - IBM Regatta
  - IBM Blue Gene/L
  - Best processor performance for codes that are not "cache friendly"
  - Good communication performance
  - Simpler programming model
  - Most expensive
- Commodity processor with custom interconnect
  - SGI Altix
    - Intel Itanium 2
  - Cray XT3, XD1
    - AMD Opteron
  - Good communication performance
  - Good scalability
- Commodity processor with commodity interconnect
  - Clusters
    - Pentium, Itanium, Opteron, Alpha
    - GigE, Infiniband, Myrinet, Quadrics
  - NEC TX7
  - IBM eServer
  - Dawning
  - Best price/performance (for codes that work well with caches and are latency tolerant)
  - More complex programming model

Loosely Coupled

Courtesy of Jack Dongarra

Architecture/Systems Continuum (2)



Courtesy of Jack Dongarra

24th List: The TOP10

TOP500 SUPERCOMPUTER SITES

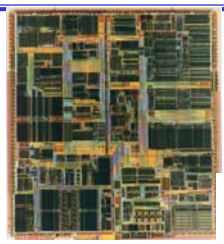
	Manufacturer	Computer	Rmax [TF/s]	Installation Site	Country	Type Year	#Proc
1	IBM	BlueGene/L β-System	136.8	Lawrence Livermore National Laboratory	USA	Custom2005	65536
2	IBM	BGW - eServer Blue Gene Solution	91.29	IBM Thomas J. Watson Research Center	USA	Custom2005	40960
3	SGI	Columbia Altix + Infiniband	51.87	NASA Ames	USA	Hybrid2004	10160
4	NEC	Earth-Simulator	35.86	Earth Simulator Center	Japan	Custom2002	5120
5	IBM	MareNostrum BladeCenter JS20, Myrinet	27.91	Barcelona Supercomputer Center	Spain	Commod2004	4800
6	IBM	eServer BG Solution	27.45	University Groningen	NL	Custom2005	12288
7	CCD	Thunder Itanium2, Quadrics	19.94	Lawrence Livermore National Laboratory	USA	Commod2004	4096
8	IBM	eServer BG Solution	18.20	EPFL	Swiss	Custom2005	8192
9	IBM	eServer BG Solution	18.20	Japan Adv. Inst. of Science and Technology	Japan	Custom2005	8192
10	Cray Inc	Red Storm, Cray XT3	15.25	Sandia National Lab	USA	Hybrid2005	5000

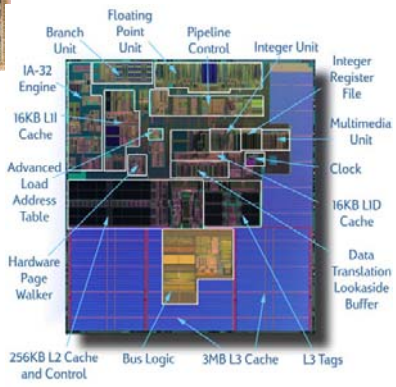
Courtesy of Jack Dongarra

All 500 systems > 1 TFlop/s; 304 machines clusters; top10 average 16K proc; US has 294 IBM 259. HP 131, SGI 24, Dell 21 and Cray 16 (6 of 10 IBM, 5 in USA); > 4800 processors

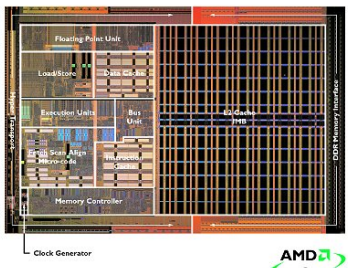
Commodity Processors

- Intel Pentium Nocona
  - 3.6 GHz, peak = 7.2 Gflop/s
  - Linpack 100 = 1.8 Gflop/s
  - Linpack 1000 = 4.2 Gflop/s
- Intel Itanium 2
  - 1.6 GHz, peak = 6.4 Gflop/s
  - Linpack 100 = 1.7 Gflop/s
  - Linpack 1000 = 5.7 Gflop/s
- AMD Opteron
  - 2.6 GHz, peak = 5.2 Gflop/s
  - Linpack 100 = 1.6 Gflop/s
  - Linpack 1000 = 3.9 Gflop/s





McKinley microprocessor



Courtesy of Jack Dongarra

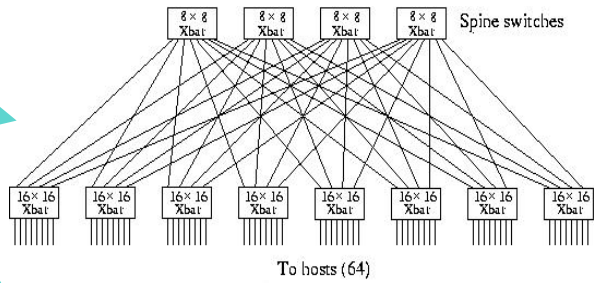
Commodity Interconnects

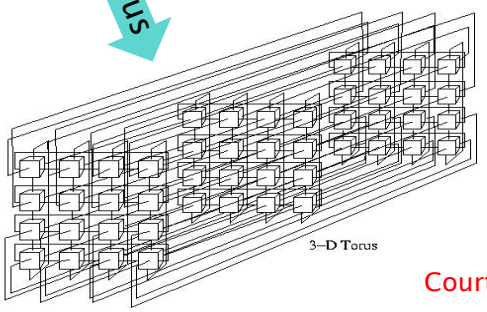
- Gig Ethernet
- Myrinet
- Infiniband
- QsNet
- SCI

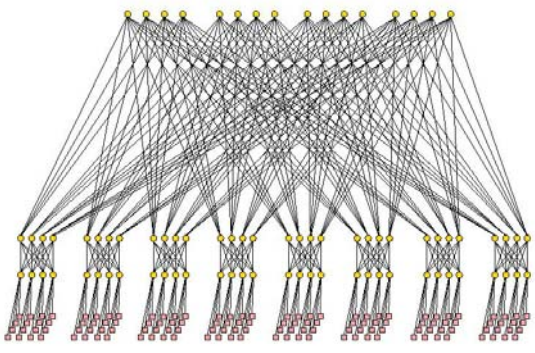
Clos

Fat tree

Torus

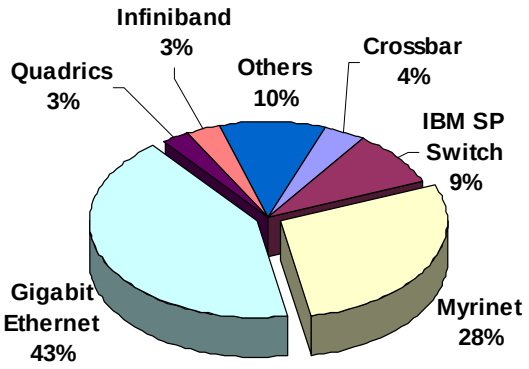






Courtesy of Jack Dongarra

Interconnects – June 2005 Top500



	Switch topology	Cost NIC	Cost Sw/node	Cost Node	MPI Lat / 1-way / Bi-Dir (us) / MB/s / MB/s
Gigabit Ethernet	Bus	$ 50	$ 50	$ 100	30 / 100 / 150
SCI	Torus	$1,600	$ 0	$1,600	5 / 300 / 400
QsNetII (R)	Fat Tree	$1,200	$1,700	$2,900	3 / 880 / 900
QsNetII (E)	Fat Tree	$1,000	$ 700	$1,700	3 / 880 / 900
Myrinet (D card)	Clos	$ 500	$ 300	$ 800	3.2 / 240 / 480
Myrinet (E card)	Clos	$ 900	$ 600	$1,400	2.6 / 450 / 900
Myrinet (F card)	Clos	$ 600	$ 300	$ 900	2.6 / 240 / 480
IB 4x	Fat Tree	$1,000	$ 400	$1,400	6 / 820 / 790

Courtesy of Jack Dongarra

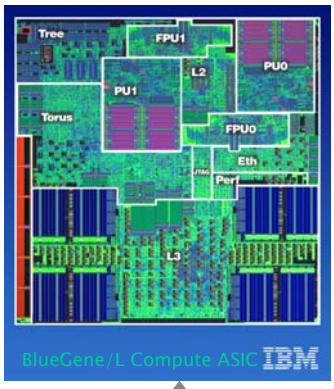
In Italy – 11 Computers on TOP500

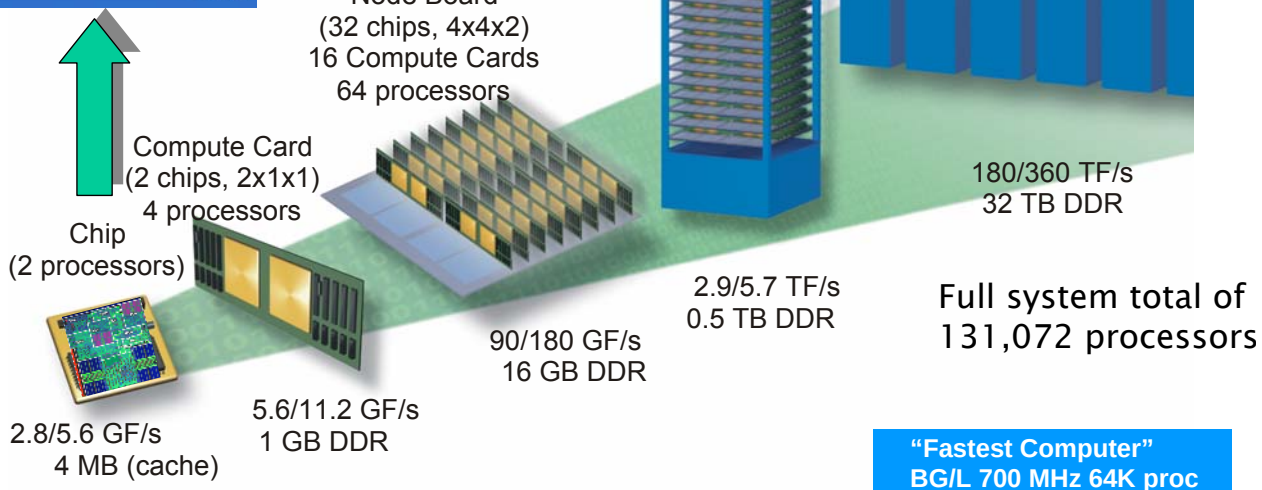
Rank	Company	System	Place	# Procs
101	IBM	eServer pSeries p5 575 1.9 GHz	CINECA	512
106	IBM	xSeries, Xeon 3.06 GHz, Myrinet	CINECA	1024
144	IBM	eServer, Opteron 2.0 GHz, Myrinet	Automotive Manufacturer	768
206	HP	Cluster 3000 DL360G3 Xeon 3.2 GHz, GigE	Energy Company	516
337	HP	SuperDome 1 GHz/HPlex	Telecom Italia	640
385	HP	SuperDome 875 MHz/HyperPlex	Hutchison H3G	704
408	HP	Integrity Superdome, 1.5 GHz, HPlex	Manufacturing Company	288
421	HP	SuperDome 875 MHz/HyperPlex	Telecom Italia	640
422	HP	SuperDome 875 MHz/HyperPlex	Telecom Italia	640
423	HP	SuperDome 875 MHz/HyperPlex	Telecom Italia	640
424	HP	SuperDome 875 MHz/HyperPlex	Telecom Italia	640

Courtesy of Jack Dongarra

IBM BlueGene/L

131,072 Processors (#1-64K and #2-40K)





System
(64 racks, 64x32x32)
131,072 procs

Rack
(32 Node boards, 8x8x16)
2048 processors

Node Board
(32 chips, 4x4x2)
16 Compute Cards
64 processors

Compute Card
(2 chips, 2x1x1)
4 processors

Chip
(2 processors)

2.8/5.6 GF/s
4 MB (cache)

5.6/11.2 GF/s
1 GB DDR

90/180 GF/s
16 GB DDR

2.9/5.7 TF/s
0.5 TB DDR

180/360 TF/s
32 TB DDR

Full system total of
131,072 processors

The compute node ASICs include all networking and processor functionality. Each compute ASIC includes two 32-bit superscalar PowerPC 440 embedded cores (note that L1 cache coherence is not maintained between these cores).
(10K sec about 3 hours; n=1.2M

Courtesy of Jack Dongarra

"Fastest Computer"
BG/L 700 MHz 64K proc
32 racks
Peak: 184 Tflop/s
Linpack: 135 Tflop/s
73% of peak