"Non è degno di uomini eccellenti
perdere ore come schiavi
nell'attività manuale di calcolare,
che potrebbe essere sicuramente
demandata ad una macchina."

Eottfried W. Leibniz (1647-1716)

Capitolo 2

Le architetture parallele

2.1 Classificazione delle architetture parallele

Una classificazione ormai standard delle architetture parallele[1] è quella ad opera di Flynn (1966). La classificazione si basa sulla nozione di flusso di *"informazione"*, dove per *"informazione"* si intende una istruzione o un dato. In Fig. 2.1 è rappresentata tale classificazione. Sulle colonne è riportato il tipo di flusso di istruzioni. Sulle righe è riportato, invece, il tipo di flusso di dati. Il flusso è *"singolo"* se viene elaborata una sola istruzione o un solo dato, è *"multiplo"* se vengono elaborate piú istruzioni o piú dati.

Nella tipologia SISD (**S**ingle **I**nstruction **S**ingle **D**ata) rientra il calcolatore mono processore. Esso ha un'unica unità di calcolo e di controllo e prevede un unico flusso di dati su cui viene eseguito un unico flusso di istruzioni. Le architetture SIMD (**S**ingle **I**nstruction **M**ultiple **D**ata) hanno una singola unità di controllo e piú ALU per eseguire una stessa istruzione su flussi di dati diversi. Le architetture MISD (**M**ultiple **I**nstructions **S**ingle **D**ata) si riferiscono a calcolatori in grado di eseguire flussi di istruzioni diverse su uno

[1] La classificazione delle architetture parallele descritta in questo paragrafo è volutamente semplificata perché è essenzialmente rivolta a evidenziare gli aspetti funzionali che interessano la computazione parallela.

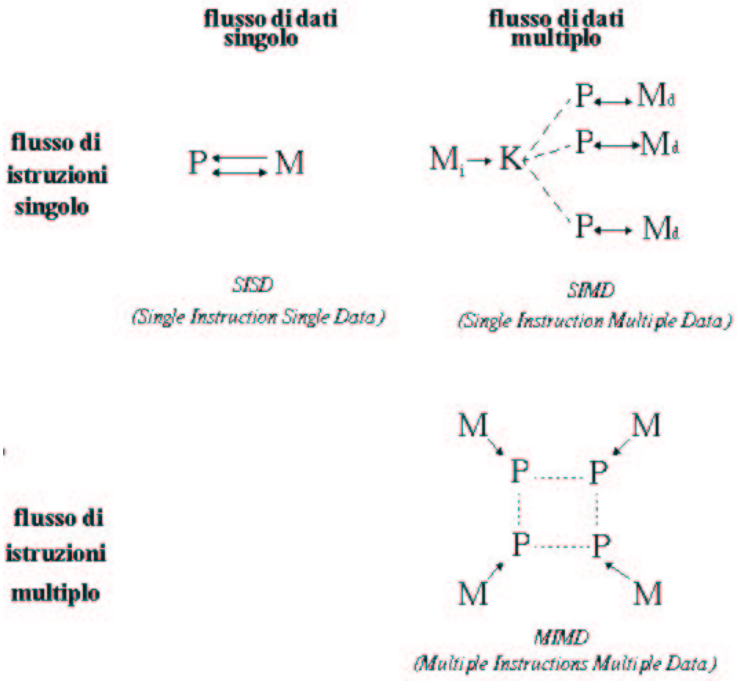

Figura 2.1: *Tassonomia di Flynn. P=unità di calcolo M=memoria*

Figura 2.2: *Dettaglio sulla tassonomia di Flynn.*

stesso insieme di dati. Alcuni considerano le macchine pipeline come MISD [51, 4]. Infine nella tipologia MIMD (**M**ultiple **I**nstructions **M**ultiple **D**ata) rientrano le architetture costituite da piú CPU che consentono l'esecuzione di flussi di istruzioni diverse su flussi di dati differenti.

In un'architettura SIMD il parallelismo è realizzato a livello di unità funzionale (ALU): piú unità aritmetico-logiche operano sotto un controllo comune, come mostrato in Fig. 2.3, eseguendo concorrentemente la stessa istruzione su dati diversi. Un esempio di architettura SIMD è la Connection Machine 2 della Thinking Machine Corporation [49].

Di nuovo, tornando all'analogia della costruzione di una casa (Capitolo 1), si può pensare a piú muratori che eseguono contemporaneamente la stessa azione su mattoni diversi (**parallelismo spaziale**) (Fig. 2.4).

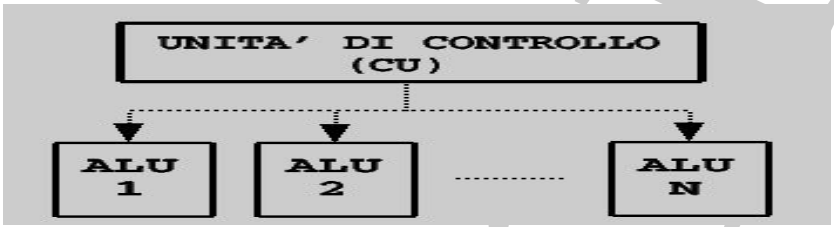

Figura 2.3: *Schema funzionale di una macchina SIMD*

Figura 2.4: *Più operai eseguono contemporaneamente la stessa azione su mattoni diversi (parallelismo spaziale)*

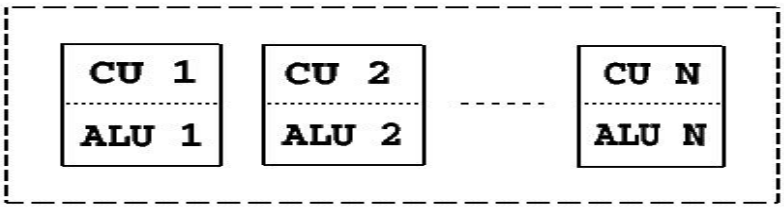

Figura 2.5: *Schema funzionale di una macchina MIMD*

Figura 2.6: *Più operai eseguono contemporaneamente azioni diverse su parti diverse (parallelismo asincrono)*

In un'architettura MIMD il parallelismo è realizzato a livello di CPU (vedi Fig. 2.5): piú CPU eseguono le proprie istruzioni su dati diversi. Esempio di sistemi MIMD sono l'SP dell'IBM, il CRAY T3D della Cray Research e l'Intel iPSC.

È come se piú operai eseguissero contemporaneamente azioni diverse su parti diverse (**parallelismo asincrono**) (vedi Fig. 2.6).

Negli anni successivi alla classificazione ad opera di Flynn, si sono realizzate nuove architetture parallele, caratterizzate dall'essere combinazioni delle due tipologie fondamentali, MIMD e SIMD. Esaminiamo alcune delle nuove architetture.

Le *SMP* (**S**ymmetric **M**ulti **P**rocessor) sono costituite da piú processori che operano indipendentemente accedendo alla stessa memoria globale; ogni processore può avere anche una propria memoria (cash), mentre, l'accesso alla memoria condivisa avviene tramite bus o switch.
Nel modello *DSM* (**D**istributed **S**hared **M**emory), invece, la memoria è fisicamente distribuita a livello hardware, ma è globalmente condivisa dal punto di vista software ovvero a livello delle operazioni di accesso (scrittura, lettura) cosí che all'utente il sistema si presenta con un'unica memoria.
Particolare attenzione merita il *cluster* [1][a], sistema costituito da calcolatori differenti collegati tra di loro attraverso una rete.
L'idea che ha portato allo sviluppo di tale tipo di sistema ha iniziato a diffondersi negli anni '60 ad opera dell'IBM. L'idea era di collegare tra loro grandi *mainframes* per ottenere sistemi di calcolo ad alte prestazioni piú economici.
Tuttavia, i sistemi cluster hanno iniziato a diffondersi negli anni '80 anche in virtú del miglioramento delle prestazioni dei microprocessori, delle reti di connessione e dello sviluppo di strumenti software per il calcolo parallelo e distribuito.
I motivi per cui oggi si ha un grande interesse per i cluster sono i seguenti:

- le workstation sono sempre piú potenti a parità di costo;
- le tecnologie di rete stanno migliorando, aumentando la larghezza di banda e diminuendo il tempo di latenza;
- sono piú facili da integrare in reti già esistenti;
- sono piú economici e costituiscono un'alternativa a piattaforme di calcolo fortemente specializzate;
- possono "scalare" facilmente; la capacità di un nodo può essere aumentata aggiungendo memoria e processori.

I cluster costruiti utilizzando componenti hardware economiche e facilmente reperibili (COTS Commodity off the shelf) e software *free* svolgono un ruolo fondamentale nella definizione di *supercalcolatore* parallelo del tipo Beowulf.

[a] Cluster, Network of Workstation (NOW) e Cluster of Workstation (COW) sono sinonimi.

I computer che fanno parte di un cluster sono generalmente connessi mediante una rete LAN (**L**ocal **A**rea **N**etwork). Le reti LAN hanno tipicamente un'alta latenza e una bassa larghezza di banda. Un ruolo fondamentale, in questi sistemi, è svolto dallo switch di interconnessione.
I *sistemi distribuiti* sono sistemi molto simili ai cluster. A differenza di questi, però, sono distribuiti su aree geografiche piú estese e coinvolgono distanze dell'ordine delle migliaia di chilometri (WAN, **W**ide **A**rea **N**etwork).
Come naturale evoluzione dei sistemi distribuiti sta emergendo il *Grid Computing* [22], la tecnologia di calcolo del XXI secolo. L'idea alla base del Grid Computing è di gestire le risorse di calcolo (hw e sw) e le applicazioni condivise tra organizzazioni diverse in maniera dinamica. La condivisione riguarda soprattutto l'accesso diretto alle macchine, al software, ai dati e ad altre risorse (strumenti di acquisizione dati, database, etc. ...). Questa condivisione deve essere necessariamente controllata e l'obiettivo è quindi di realizzare protocolli, servizi e strumenti che consentano una condivisione delle risorse sicura, trasparente ed efficiente tra le organizzazioni.

Un'ulteriore classificazione delle architetture multiprocessore è legata alla modalità di accesso della CPU alla memoria. Distinguiamo, quindi, i sistemi multiprocessori a memoria condivisa (*shared memory*) da quelli a memoria distribuita (*distributed memory*).

Nei sistemi a memoria condivisa le CPU condividono la stessa memoria come mostrato in Fig. 1.7. In questo caso esiste un'unica memoria (eventualmente fisicamente distribuita) che dal punto di vista funzionale è condivisa da tutti i processori.

Il calcolatore della SGI-CRAY Power Challenge è un sistema a memoria condivisa.

In un sistema a memoria distribuita (Fig. 1.8), invece, la memoria è associata ad un singolo processore e ognuno di questi può indirizzare solo la propria memoria. Se un processore richiede un dato risiedente nella memoria di un altro processore, è necessario attivare un trasferimento del dato dalla memoria di un processore a

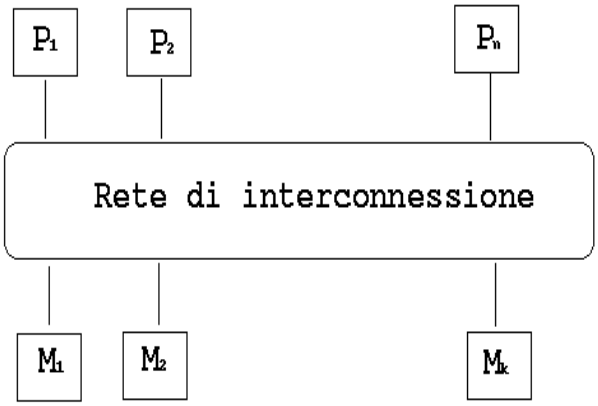

Figura 2.7: *Esempio di multiprocessore UMA.*

quella dell'altro.

Il calcolatore SP dell'IBM e l'Intel iPSC sono esempi di sistemi a memoria distribuita, cosí come i cluster.

La rete di interconnessione influisce in maniera considerevole sulle prestazioni del sistema. Se questo tempo è identico per tutti gli accessi alla memoria il calcolatore prende il nome di multiprocessore con accesso uniforme alla memoria (UMA, **U**niform **M**emory **A**ccess) (vedi Fig. 2.7), al contrario, se il tempo di accesso alla memoria non è lo stesso, il sistema prende il nome di multiprocessore con accesso non uniforme (NUMA, **N**on **U**niform **M**emory **A**ccess) (vedi Fig. 2.8). Un esempio di architettura Shared Memory di tipo UMA è quella di un **S**ymmetric **M**ulti **P**rocessor (**SMP**). Un esempio di architettura Shared Memory di tipo NUMA è quella di un calcolatore **D**istributed **S**hared **M**emory (**DSM**).

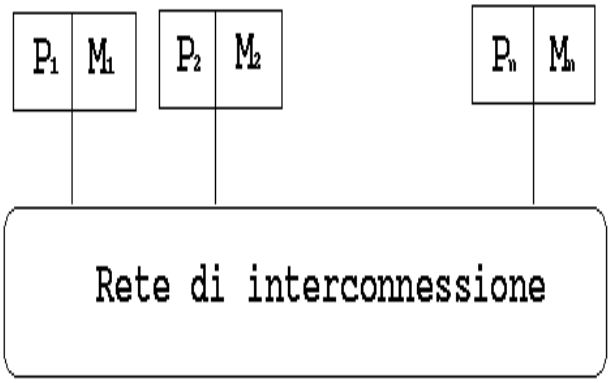

Figura 2.8: *Esempio di multiprocessore NUMA.*

La funzione dell'unità di memoria è contenere programmi e dati. Si distinguono due tipi di dispositivi di memoria, quella principale[a] e quella secondaria. La capacità massima della memoria è determinata dallo schema di indirizzamento. Un calcolatore con indirizzi a k bit è in grado di indirizzare fino a 2^k locazioni di memoria. Il numero di locazioni rappresenta la dimensione dello spazio di indirizzamento del calcolatore. I programmi risiedono nella memoria principale durante l'esecuzione: l'istruzione e i dati vengono scritti o letti dalla memoria, sotto il controllo diretto del processore. Il tempo necessario per accedere ad una world costituita da $m \cdot locazioni$ è detto *tempo di accesso* (T_a) e può essere espresso come:

$$T_a = t_l + m \cdot t_{tw}$$

dove con t_l si intende il tempo di latenza, equivalente al tempo di start up di una comunicazione[b], ed è dovuto solo ai tempi hw e sw di connessione di tutte le componenti coinvolte nella comunicazione; t_{tw} rappresenta il tempo di trasferimento di una locazione, mentre m è il numero complessivo di locazioni che costituiscono la *world*. Nella maggior parte dei calcolatori attuali, il tempo di accesso varia da 10 a 100 nsec [(c)].

[a] La memoria principale è quella che nello schema funzionale della macchina di Von Neumann è direttamente collegata alla CPU.

[b] Anche tempo di comunicazione di un messaggio vuoto.

[c] Un processore Pentium IV, ad esempio, con una frequenza di clock di 1.5 GHz, e con 512 M di RAM ha due livelli di cache, $L1$ di $8K$ ed $L2$ di $256K$

Il trasferimento dei dati fra la memoria e la CPU avviene mediante l'utilizzo di due registri della CPU, chiamati MAR (**M**emory **A**ddress **R**egister), registro degli indirizzi della memoria, e MDR (**M**emory **D**ata **R**egister), registro dei dati della memoria. Se il registro MAR contiene k bit e il registro MDR ne contiene n, allora l'unità di memoria può contenere fino a 2^k locazioni indirizzabili e durante un *"ciclo di memoria"* un dato di n bit viene trasferito tra memoria e CPU. Questo trasferimento avviene mediante il bus del processore che ha k linee degli indirizzi. Il bus comprende anche le linee di controllo per la lettura, la scrittura e il segnale di completamento della funzione di memoria (**M**emory **F**unction **C**ompleted, MFC) che servono per coordinare i trasferimenti dei dati. Negli ultimi venti anni si è assistito ad una considerevole crescita delle prestazioni dei microprocessori.
Mediamente, la velocità di un microprocessore è raddoppiata ogni 12-18 mesi[a]. D'altra parte la velocità dell'accesso alla memoria non ha avuto lo stesso tasso di crescita facendo cosí aumentare il *gap* tra le prestazioni dei processori e quella degli accessi in memoria (Fig. 2.9).
Poiché ogni locazione della memoria principale deve essere direttamente accessibile in un tempo brevissimo, molti calcolatori hanno memorie secondarie piú lente, piú economiche e di solito anche piú grandi. Le memorie secondarie sono usate per contenere insiemi di dati molto piú grandi di quanto la memoria centrale stessa possa contenere. Esiste una vasta gamma di dispositivi di memoria secondaria che include dischi magnetici, nastri, floppy disk, CD ROM.
Uno schema logico dei diversi tipi di memorie presenti in un calcolatore è descritto in Fig. 2.10. Il livello piú alto è costituito dalla memoria a piú rapido accesso, ma a costo/bit maggiore. Subito dopo si trova la memoria cache. In un sistema dotato di memoria cache, l'esecuzione di un programma avviene trasferendo istruzione e dati alla memoria cache appena sono necessari. Questa operazione viene eseguita ad una velocità relativamente lenta. Dopo questo trasferimento, successive richieste delle stesse istruzioni e dati vengono soddisfatte dalla cache: tutto ciò avviene a una velocità notevolmente superiore. Poichè l'informazione contenuta nella cache viene utilizzata ripetutamente prima di venire sostituita, si ottiene un significativo miglioramento delle prestazioni.

[a]La legge di Moore [41], formulata nel 1965 da Gordon Moore, uno dei fondatori della Intel, teorizza il raddoppio ogni 12-18 mesi del numero di transistor ospitati in un chip. Essa ha caratterizzato l'evoluzione dell'industria del personal computer ed è considerata valida ancora oggi. Seguendo il ritmo della legge, in un solo chip si è addensato un numero sempre crescente di componenti: nel 1964 un chip conteneva 32 elementi; nel 1974 65mila, oggi circa 28 milioni. Come conseguenza la potenza e la velocità dei calcolatori sono andate aumentando in modo esponenziale e allo stesso tempo le dimensioni dei singoli componenti sono diminuite: oggi la media è di 180 nanometri, in altre parole milionesimi di millimetro. È importante sottolineare, comunque, che, proprio grazie al parallelismo implementato nelle componenti hw degli attuali microprocessori, la Legge di Moore continua ad essere ancora vera.

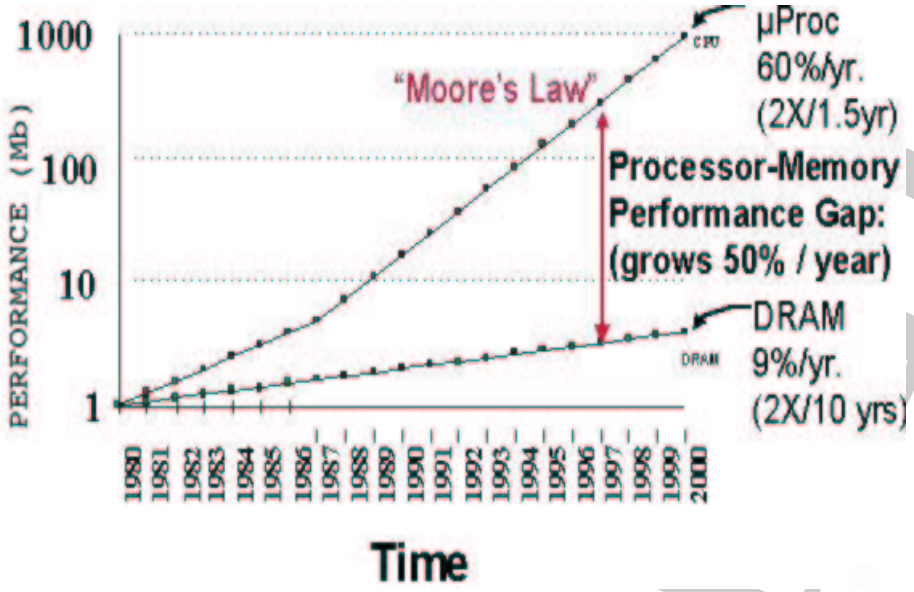

Figura 2.9: *Mentre le prestazioni dei microprocessori raddoppiano ogni 12-18 mesi (legge di Moore), la velocità degli accessi in memoria raddoppia ogni dieci anni. Ogni anno il gap processore/memoria aumenta del 50% (http://www.netlib.org/utk/people/JackDongarra).*

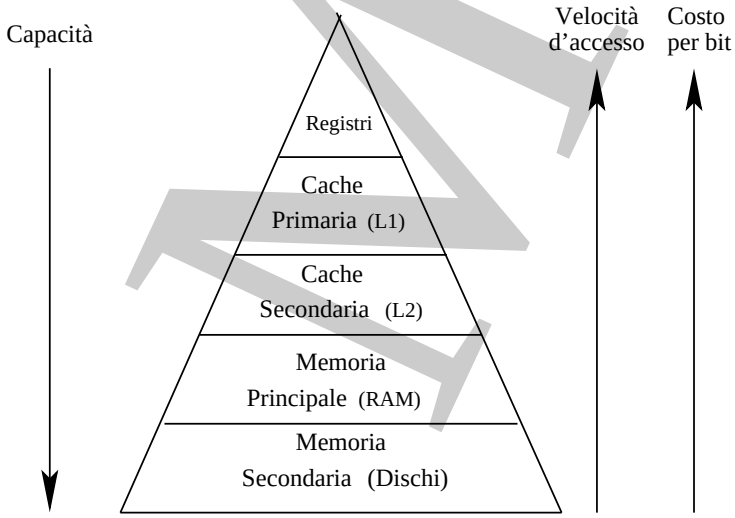

Figura 2.10: *Gerarchia della memoria. Le frecce indicano il verso crescente della capacità, della velocità di accesso e del costo per bit delle memorie.*

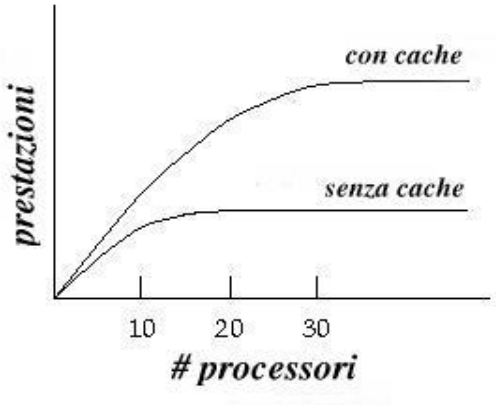

Figura 2.11: *Evoluzione della performance (misurata in Mflops) di un sistema di calcolo con e senza memoria cache.*

2.2 Reti di interconnessione per sistemi a memoria distribuita

In questo paragrafo esaminiamo alcune possibili realizzazioni di una rete di interconnessione tra nodi, intendendo per nodo il sistema cosituito da CPU e memoria. Il tipo di connessione è detto *topologia* del calcolatore.

In generale, il collegamento deve consentire il trasferimento di informazioni tra le componenti del sistema. Il traffico nella rete è costituito dal trasferimento di dati e istruzioni.

Il tipo di rete da adottare viene valutato in base al costo, alla larghezza di banda, alla quantità effettiva di informazioni trasmesse nell'unità di tempo e dalla faciltà di realizzazione. Il termine *larghezza di banda* (bandwidth) fa riferimento alla velocità di un collegamento di trasferimento dati ed è espressa in bit o byte per secondo. La quantità effettiva di informazione trasmessa nell'unità di tempo (*effective throughput*) è la velocità effettiva di trasferimento dei dati.

Possono essere utilizzate topologie differenti per realizzare la rete di interconnessione. Vediamone alcune.

- **Reti ad anello**
  Una delle topologie di rete piú semplice utilizza un anello per collegare i nodi del sistema, come mostrato in Fig. 2.12.

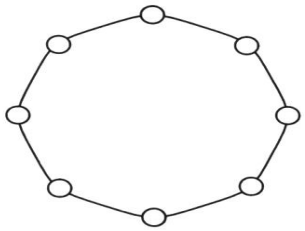

Figura 2.12: *Topologia ad anello (ring)*

Il vantaggio principale di questa organizzazione è costituito dal fatto che l'anello è di facile realizzazione. I collegamenti dell'anello possono essere ampi. Tuttavia, non è utile costruire un anello molto lungo per collegare molti nodi, poiché la latenza del trasferimento di informazioni diverebbe troppo elevata.

- **Connessione totale**
  In questo tipo di topologia ogni nodo ha un legame diretto con tutti gli altri nodi (Fig. 2.13).

  Una connessione completa di n nodi fornisce un'ampiezza di banda proporzionale ad n^2. Infatti, essendo ogni nodo collegato a tutti gli altri nodi si hanno $n(n-1)$ collegamenti e tutti possono comunicare con tutti. Sfortunatamente anche il costo

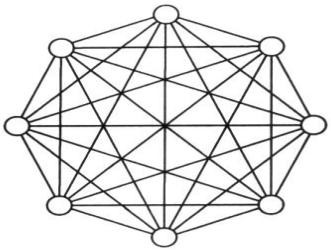

Figura 2.13: *Topologia a comunicazione totale (total connection)*

cresce proporzionalmente ad n^2, rendendola inadatta per grossi sistemi.

- **Reti a ipercubo**
  È un cubo a n dimensioni che collega 2^n nodi (Fig. 2.14). Gli archi del cubo rappresentano collegamenti di comunicazione bidirezionali tra nodi adiacenti. In un cubo n-dimensionale, ogni nodo è collegato direttamente ad n nodi vicini.

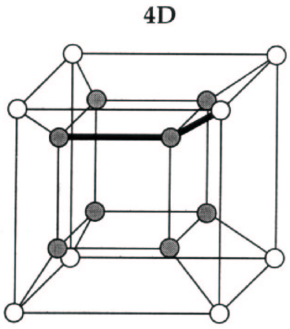

Figura 2.14: *Topologia ad ipercubo*

Tali reti furono utilizzate per numerosi sistemi utilizzando ti-

picamente una trasmissione seriale. Esempi di tali macchine sono iPSC della Intel, NCUBE/10 della NCUBE e CM-2 della Thinking Machines.

- **Reti a matrice (mesh)**
  Uno dei modi piú naturali di collegare un numero elevato di nodi è l'impiego di una topologia simile ad una matrice, o mesh, come quella rappresentata in Fig. 2.15. Ogni nodo costituisce un elemento della matrice e sono collegati a due a due i nodi adiacenti. Anche in questo caso i collegamenti tra due nodi sono bidirezionali.

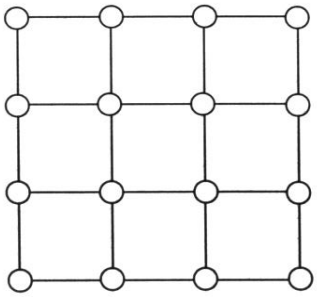

Figura 2.15: *Topologia mesh*

L'instradamento in una rete a matrice è effettuato selezionando il percorso tra un nodo sorgente e un nodo destinatario in modo tale che il trasferimento avvenga prima nella direzione orizzontale. Una volta raggiunta la colonna in cui è presente il nodo destinatario, il trasferimento procede in direzione verticale lungo questa colonna.

Se si effettua una connessione periodica tra i nodi appartenenti a bordi opposti del mesh, il risultato è una rete che comprende

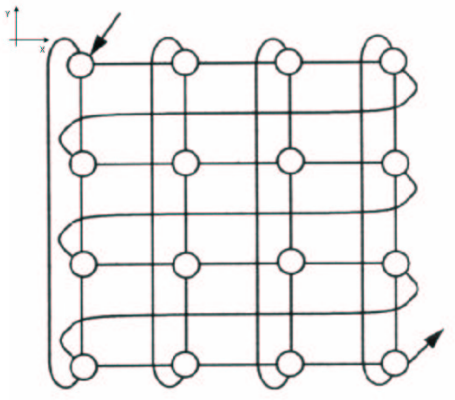

Figura 2.16: *Topologia toroidale*

un insieme di anelli bidirezionali nella direzione x (orizzontale) connesso a un insieme di anelli nella direzione y [2] (verticale). Questa rete è chiamata toro ed è rapprentata in Fig. 2.16.

- **Le reti Omega**
  Le reti omega (Fig. 2.17) utilizzano interruttori 2×2 [3]. L'interruttore ha due input e due output. I messaggi che arrivano sulle due linee in input possono essere commutati sulle due linee in output. Per n CPU ed n memorie sono necessari $log_2 n$ stadi, con $n/2$ interruttori per stadio, per un totale di $(n/2)log_2 n$ interruttori.

  Per comunicare con un modulo di memoria, la CPU manda un messaggio all'interruttore al primo stadio contenente l'indirizzo del modulo. All'i-esimo stadio l'interruttore prende l'i-esimo bit e lo usa per l'instradamento.

[2] Le direzioni a cui si fa riferimento sono quelle di un sistema ortogonale $x0y$.
[3] Le reti Omega sono alla base della connessione tra i nodi della Butterfly.

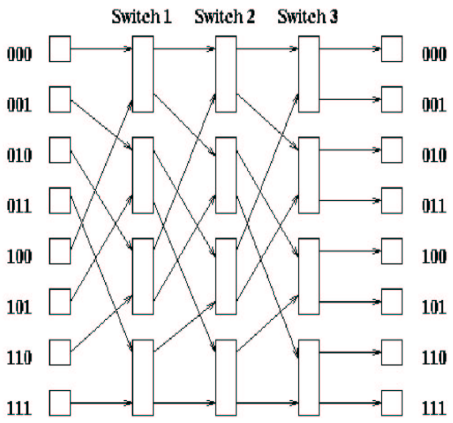

Figura 2.17: *Rete omega*