

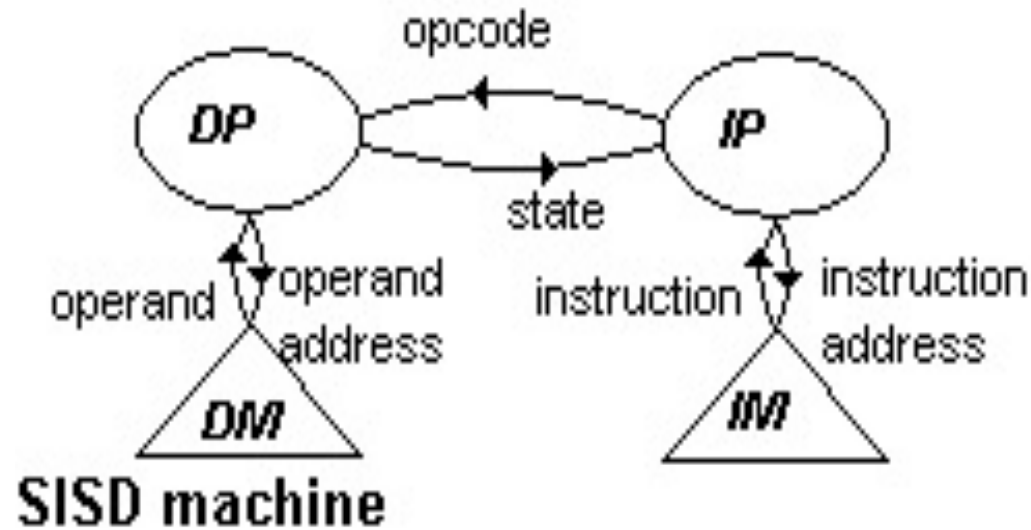
Tassonomia di Flynn

	<i>SI</i> <i>(Single Instruction stream)</i>	<i>MI</i> <i>(Multiple Instruction stream)</i>
<i>SD</i> <i>(Single Data stream)</i>	Macchine SISD	Macchine MISD
<i>MD</i> <i>(Multiple Data stream)</i>	Macchine SIMD	Macchine MIMD

Classifica un sistema di elaborazione da 2 punti di vista:

- in base alla capacità di avere più flussi di istruzioni
- in base alla capacità di avere più flussi di dati

SISD

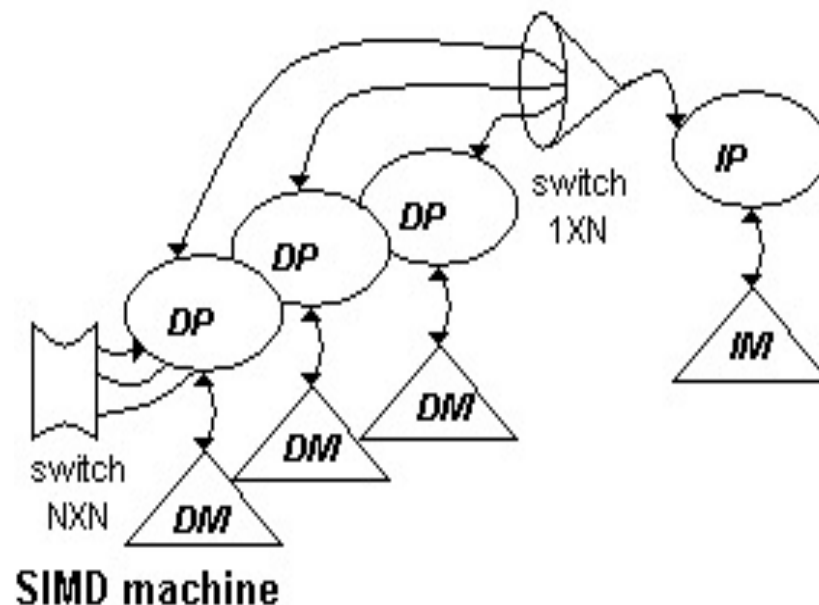


Tradizionale macchina sequenziale (o di **Von Neumann**) usata da tutti i calcolatori convenzionali



Un'unica istruzione è eseguita a ogni step temporale

SIMD



più unità di elaborazione eseguono contemporaneamente la stessa istruzione, lavorando però su flussi di dati differenti

topologie di interconnessione regolari o create *ad hoc* (i.e., in base alla struttura del problema)

comunicazioni regolari (cioè che rispettano la topologia fisica) non creano conflitti, sono efficienti e, dunque, poco costose

A large orange circle with the text 'SIMD' in white. A small blue circle is at the bottom left of the orange circle. In the top right corner, there are several yellow curved lines of varying lengths.

SIMD

Modello di computazione sincrono (1 unità di controllo)

Parallelismo temporale. Pipeline: fasi diverse di un'unica istruzione sono eseguite in parallelo in differenti moduli connessi in cascata

Parallelismo spaziale. I medesimi passi sono eseguiti contemporaneamente su un array di processori perfettamente uguali, sincronizzati da un solo controllore

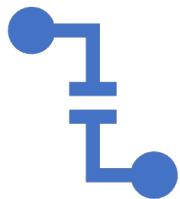


SIMD

Esempi:

- Supercomputer vettoriali, usati per applicazioni dove si lavora su grandi matrici
- Vector Processor con caratteristiche pipeline (parallelismo temporale)
- Array Processor (parallelismo spaziale)
- Systolic Array (parallelismo temporale/spaziale)

SIMD - Vector Processor



Diversità di funzionamento tra un processore scalare ed un processore vettoriale

Esempio

`c = a + b;`

Processore scalare: gli operandi sono scalari

Processore vettoriale: gli operandi sono vettori



Compilatore vettoriale

Esempio

`int i = 0;`

`for (; i < 10; i++)`

• `c[i] = a[i] + b[i];`

Riconosce tutti quei cicli sequenziali trasformabili in un'unica operazione vettoriale

MISD

Più flussi di istruzioni
lavorano
contemporaneamente su
un unico flusso di dati

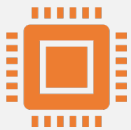


Categoria praticamente
vuota (pipeline?)

MIMD

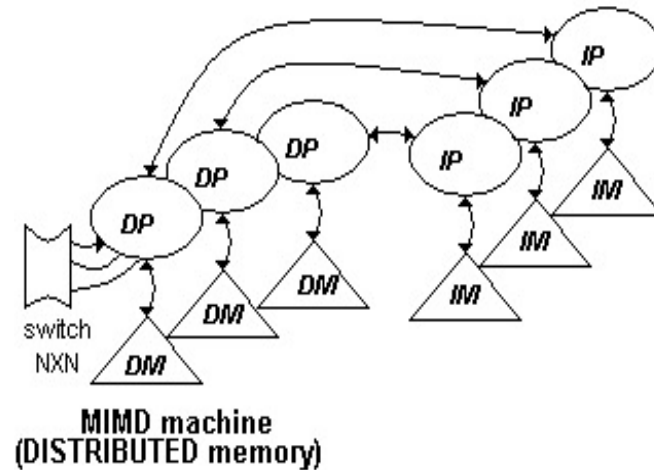


Più unità di elaborazione



Più flussi di istruzioni sono in esecuzione contemporaneamente su più processori, elaborando insiemi di dati distinti, privati o condivisi

DM-MIMD (a memoria distribuita)



Ogni coppia *IP-DP* (e relative memorie) costituisce in pratica una macchina SISD

Tra i nodi non esiste memoria condivisa e ogni nodo esegue indipendentemente un flusso di istruzioni su un differente insieme di dati, memorizzati su spazi differenti

La comunicazione è realizzata mediante una sottorete dedicata (*switch NxN*)

DM- MIMD Esempi

Reti di calcolatori

Multicomputers

- Rete di interconnessione regolare e diretta (ipercubi, mesh, torus), attraverso cui i nodi si scambiano informazioni secondo il paradigma *message passing*
- Modello di comunicazione che si discosta dalla topologia dell'architettura
- Algoritmi ad elevata località
- Elevata scalabilità

DM-MIMD MPP (Massively Parallel Processing)

Elaborazione MPP: applicazioni scientifiche e particolari contesti di calcolo commerciale-finanziario

Sistema MPP:

- migliaia di nodi (CPU standard, ognuna con la propria memoria e la propria copia del SO)
- rete di interconnessione custom molto potente (larga banda e bassa latenza)

Affinché l'elaborazione MPP dia effettivi vantaggi occorre disporre di software capace di partizionare il lavoro e i dati su cui opera tra i vari processori




A large orange circle on the left side of the slide, partially cut off by the edge.

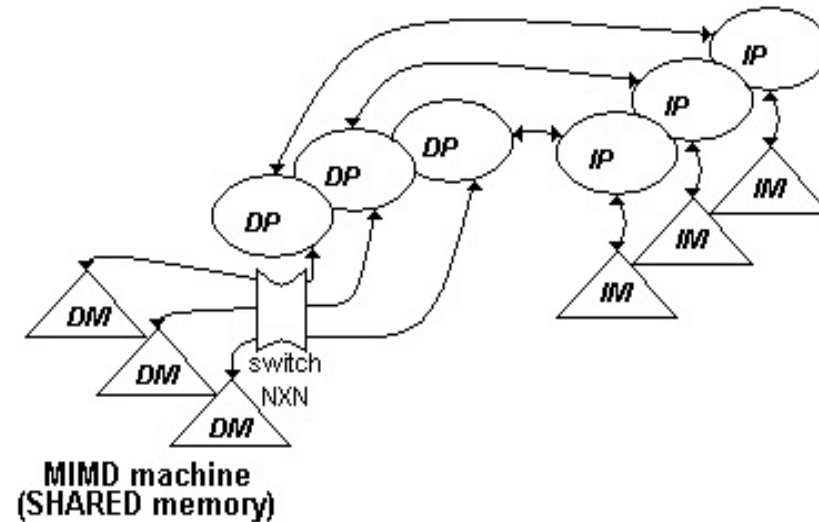
DM-MIMD Cluster Of Workstations

Connessioni: Gigabit Ethernet

Caratteristiche:

- 1.High-availability.* In caso di guasti, la computazione può migrare da un nodo all'altro
 - 2.Load-balancing.* I task da eseguire sono allocati nei nodi che hanno il minor carico
- 
- A yellow dashed line in the bottom right corner, consisting of several short, curved segments.

SM-MIMD (memoria condivisa)



Multiprocessors

Comunicazione tra processori effettuata condividendo aree di memoria

Lo *switch NxN* deve essere molto efficiente

Poiché il numero **N** di processori è “piccolo” ($N < 100$), l’accoppiamento fra i nodi può essere stretto

Limitata scalabilità

Confronto tra SIMD e MIMD

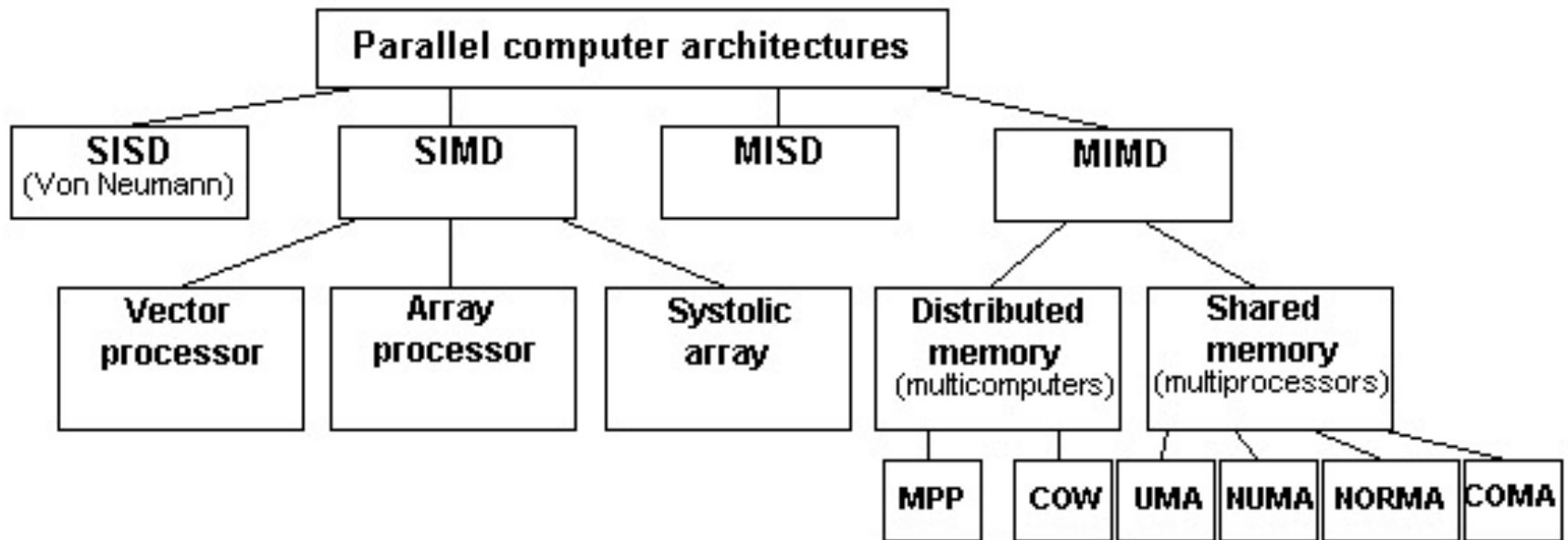
Le SIMD richiedono meno hw delle MIMD (unica Unità di Controllo)

Le MIMD usano spesso processori *general-purpose*, dunque sono meno costose delle SIMD

Le SIMD usano meno memoria delle MIMD (una sola copia del programma)

Le MIMD godono di una grande flessibilità in termini di modelli computazionali supportati

Tassonomia estesa





Topologie di interconnessione

Bus

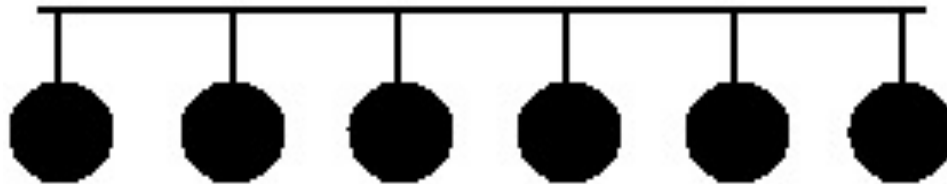
Configurazione semplice e affidabile

Grado: 1 (per tutti i nodi)

Diametro: 1

totale di link: 1

Competizione massima sull'accesso al mezzo



Linear array

Grado: per il “primo” e l’ “ultimo” nodo è 1, mentre per i restanti nodi è 2

Diametro: $N-1$

totale di link: $N-1$

Competizione ridotta al minimo

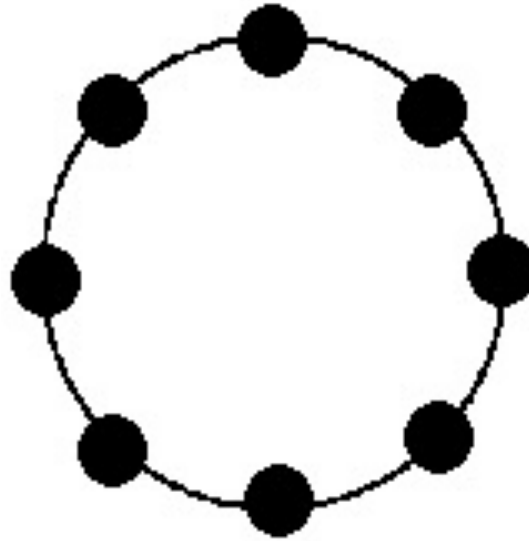
comunicazioni in contemporanea (caso ideale): $N/2$

Nodi capaci di offrire servizi di routing

No tolleranza ai guasti



Ring



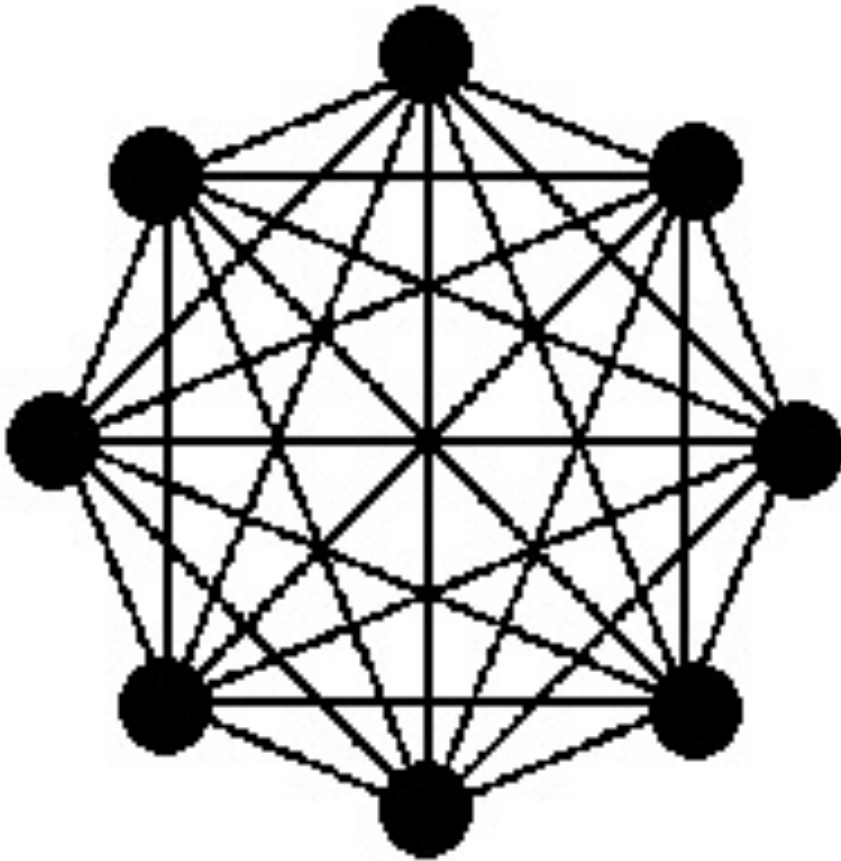
Grado: 2 (per tutti i nodi)

Diametro: $\lfloor N/2 \rfloor$

totale di link: N

Tolleranza ai guasti: 1

Connessione completa (tutti-a-tutti)

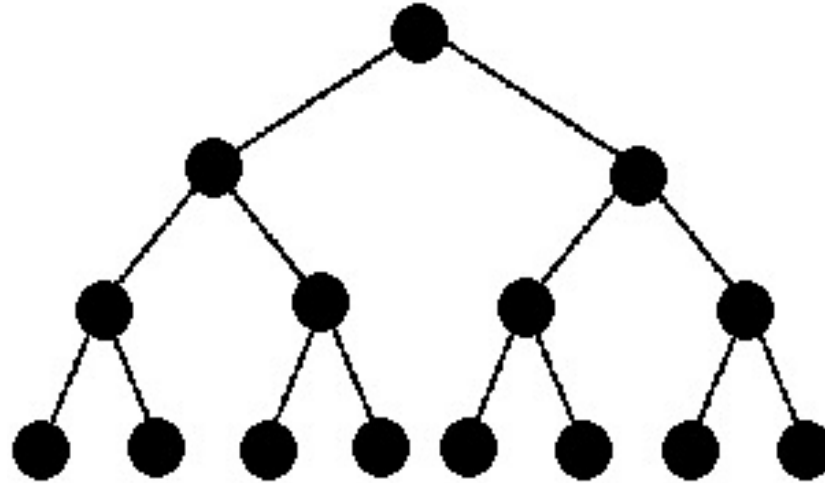


Grado: $N-1$ (per tutti i nodi)

Diametro: 1

totale di link: $N*(N-1)/2$ (non scalabile)

B-Tree



Altezza albero (h): $h = \lceil \log_2 N \rceil$

Grado: per la radice è 2; per le foglie è 1; per gli altri nodi è 3

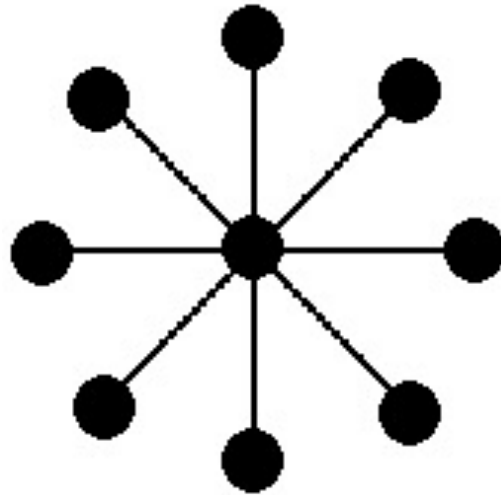
Diametro: $2 \cdot (h-1)$

totale di link: $N-1$

Rami alti congestionati (topologia non scalabile). Soluzione possibile: topologia a *fat-tree*

Radice: potenziale “punto debole”

Star



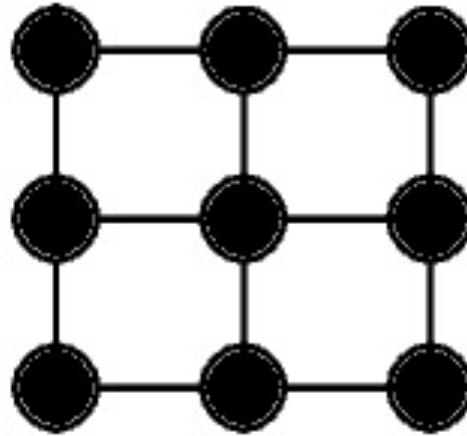
Grado: per il nodo centrale è $N-1$, mentre per gli altri nodi è 1

Diametro: 2

totale di link: $N-1$

Tolleranza ai guasti fortemente dipendente dalla “robustezza” del nodo centrale

Mesh (2-D)



r : radice quadrata di N

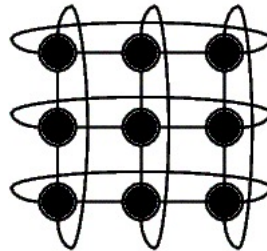
Grado: per i nodi ai vertici è 2; per i nodi “centrali” ai lati è 3; per i restanti nodi è 4

Diametro: $2*(r-1)$

totale di link: $2*N-2*r$

Resistenza ai guasti buona

Torus (2-D)



Grado: 4 (per tutti i nodi)



Diametro: $2 * \lfloor r/2 \rfloor$

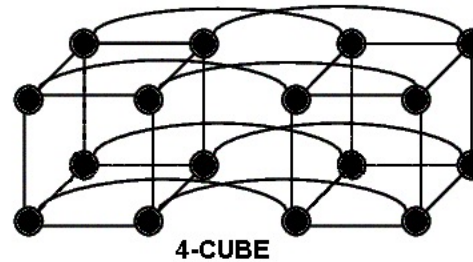
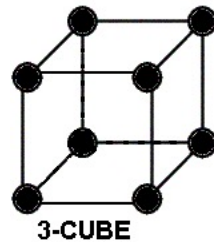


totale di link: $2 * N$



Topologia ben scalabile e notevolmente resistente ai guasti

Ipercubo (d-CUBE)



Dimensione ipercubo: d

$N: 2^d$

Grado: d (per tutti i nodi)

Diametro: $\log_2 N = d$

totale di link: $d \cdot N / 2$

Topologia scalabile solo con un numero di nodi potenza di 2

Numerazione nodi: Codice Binario di Gray

Metriche di prestazione



Speed-up ed Efficienza

Speed-up (S)

$$S = T_1/T_N$$

- Guadagno di velocità rispetto ad una esecuzione su *uniprocessore*
- Ideale: *speed-up* lineare con il numero di processori (N) usati nella macchina parallela
- Realtà: $S < N$
- Il valore dello *speed-up* dipende dalle applicazioni, ma anche dall'architettura: nelle SIMD spesso $S \approx N$, mentre nelle MIMD è difficile far crescere S (non è facile far lavorare pienamente tutte le CPU)

Efficienza (E)

$$E = S/N$$

- Misura direttamente collegata allo *speed-up*
- Ideale: $E = 1$
- Realtà: $E < 1$



Tempo sequenziale

- Tempo sequenziale (Tseq): tempo impiegato per eseguire istruzioni non parallelizzabili (operazioni di I/O, costrutti condizionali, algoritmi intrinsecamente sequenziali, ecc.)
- **Legge di Amdahl**: un parallelismo “perfetto” (nelle varie attività compiute da un calcolatore) non è **mai** raggiungibile poiché saranno **sempre** presenti sequenze di sw intrinsecamente seriale
- La **legge di Amdahl** ridefinisce lo *speed-up*:

$$S = T_1 / \{T_{seq} + [(T_1 - T_{seq})/N]\}$$

- Viene perciò posto un limite superiore per S: anche se $N \rightarrow \infty$, avremmo:

$$S = T_1 / T_{seq}$$

Esempio (algoritmo non parallelizzabile)

$$f_{n+2} = f_{n+1} + f_n \quad \text{con } f_0 = f_1 = 1 \text{ ed } n = 0, 1, 2, \dots$$

Multitasking

- Di notevole importanza pure nelle macchine parallele per mantenere lo sfruttamento delle varie CPU altissimo
- Deve rispettare il seguente vincolo:

$$P \gg N$$

