

PROGRAMMAZIONE CONCORRENTE

Introduzione e definizioni

- Un **programma concorrente** contiene *due o più processi* che lavorano assieme per eseguire una determinata applicazione.
- Ciascun processo è *un programma sequenziale*, cioè una sequenza di istruzioni eseguite una dopo l'altra.
- Un programma sequenziale ha un *single thread of control*, un programma concorrente ha *multiple threads of control*.
- I processi in un programma concorrente *comunicano* tra loro utilizzando *variabili condivise o messaggi*

- I processi devono *sincronizzarsi* gli uni con gli altri.
- Esistono due tipi base di sincronizzazione: *mutua esclusione* e *condizione di sincronizzazione*.
- La mutua esclusione garantisce ai processi *l'accesso esclusivo* ad alcune risorse; la condizione di sincronizzazione consente di *ritardare un processo* fino al verificarsi di un determinato evento.

Origini

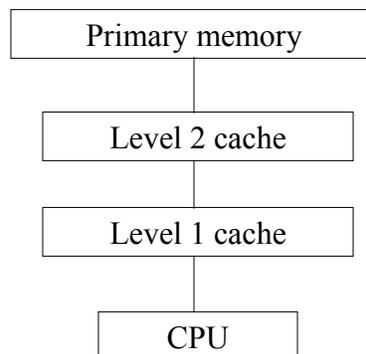
- La programmazione concorrente nasce negli anni 1960 nell'ambito dei *sistemi operativi*.
- Introduzione dei *canali o controllori di dispositivi* : consentano l'esecuzione concorrente di operazioni di I/O e delle istruzioni dei programmi eseguiti dal calcolatore centrale.
- Comunicazione tra canale ed unità centrale tramite il *segnale di interruzione*.
- Come conseguenza dell'interruzione parti di un programma possono essere eseguiti in un ordine *non predicibile*(interferenza su variabili comuni).

- Successivamente furono introdotti i *sistemi multiprocessore*. Inizialmente costosi, ora ampiamente diffusi. Macchine *massively parallel processors*.
- I sistemi multiprocessore consentono a differenti processi appartenenti alla stessa applicazione di essere *eseguiti in parallelo* e quindi all'applicazione di essere eseguita più velocemente.
- *Problemi:*
 - Come suddividere un'applicazione in processi?
 - Quanti processi utilizzare?
 - Come garantire la corretta sincronizzazione delle loro operazioni?

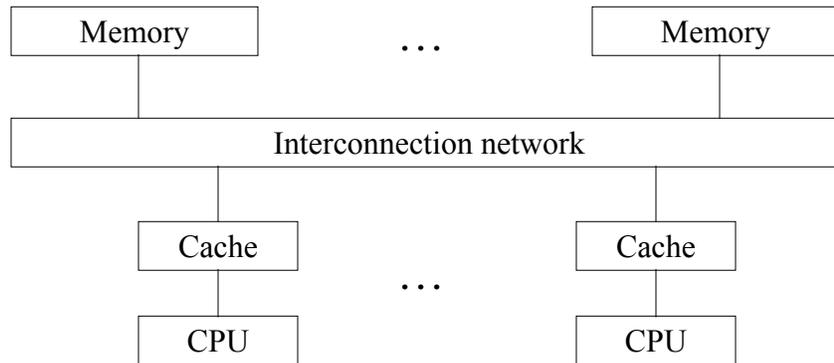
Queste decisioni dipendono dal tipo di applicazione e dal tipo di architettura hardware.

Tipi di architettura

- *Single processor*



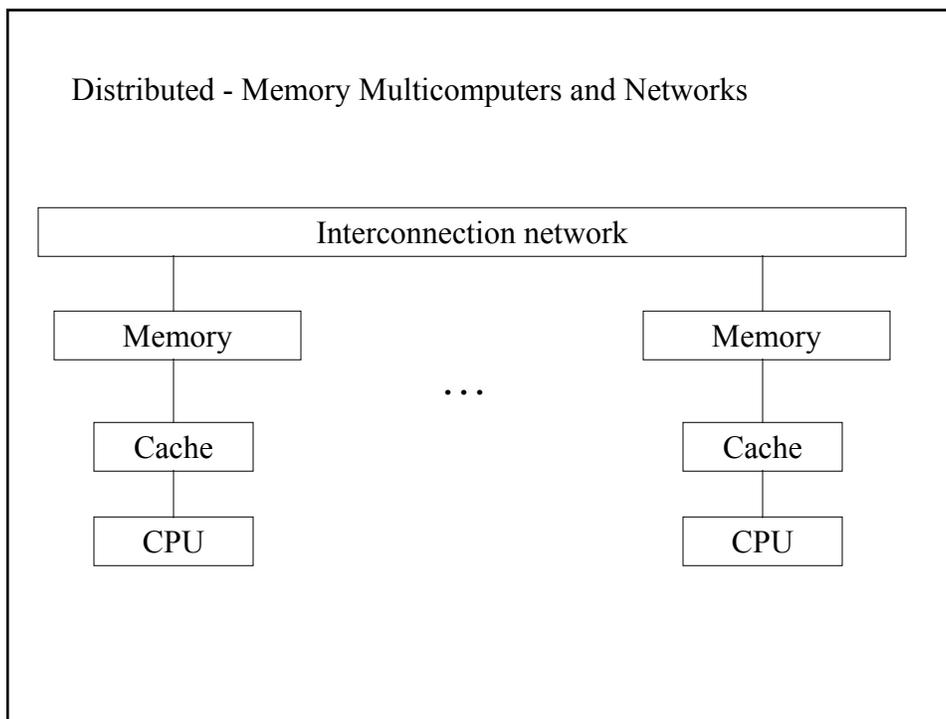
Shared- Memory Multiprocessors



In sistemi a multiprocessore con un numero ridotto di processor (2a 30), la rete di interconnessione è realizzata da un memory bus o da crossbar switch.

- In sistemi a multiprocessore con un numero ridotto di processor (da 2 a 30 circa), la rete di interconnessione è realizzata da un *memory bus* o da *crossbar switch*.
- *UMA* (Uniform Memory Access). Tempo di accesso uniforme da ogni processore ad ogni locazione di memoria. Si chiamano anche *symmetric multiprocessors* (*SMP*).
- In sistemi con un numero elevato di multiprocessori (decine o centinaia) la memoria è *organizzata gerarchicamente* (per evitare la congestione del bus). La rete di interconnessione è un insieme di switches e memorie strutturato ad albero. Ogni processore ha memorie che sono più vicine ed altre più lontane lontane *NUMA* (Non Uniform Access Time).

Distributed - Memory Multicomputers and Networks



- *Multicomputer.*

I processori e la rete sono fisicamente vicini (nella stessa struttura): *tightly coupled machine*.

La rete di interconnessione rappresenta un cammino di comunicazione tra i processori ad alta velocità e larghezza di banda (es. macchine a ipercubo).

- *Network system.*

I nodi sono collegati da una rete locale (es. Ethernet) o da una rete geografica (Internet): *Loosely coupled multiprocessors*.

- I nodi di una *distributed memory machine* possono essere o singoli processori o *shared memory multiprocessor*.

- *Distributed shared memory*: realizzazione distribuita dell'astrazione *shared memory*.

Tipi di applicazioni

a) multithreaded systems

- Applicazioni strutturate come un insieme di processi (thread) per *semplificare* la loro programmazione.
- Sono caratterizzati dal fatto che esistono *più processi che non processori* per eseguire i processi.
- I processi sono *schedulati ed eseguiti indipendentemente*.
- *Esempi di applicazioni:*
 - Sistemi a finestre su PC o Workstation
 - Sistemi operativi time-sharing e multiprocessor
 - Sistemi real time e di controllo dei processi

b) Sistemi distribuiti

- Le componenti dell'applicazione(*intrinsecamente distribuite*) vengono eseguite su macchine collegate da una rete locale rete locale o geografica
I processi comunicano scambiandosi messaggi.
- *Esempi di applicazioni*
 - File server in rete
 - Data-base systems per applicazioni bancarie etc..
 - Web server su Internet
 - Sistemi fault tolerant
- Tipica organizzazione : *client- server*.
- I componenti in un sistema distribuito sono spesso multithreaded applications

c) Elaborazioni parallele

- Obiettivo: risolvere un dato problema più velocemente(o un problema di dimensioni più elevate nello stesso tempo).
- Sono eseguite su *processori paralleli* facendo uso di algoritmi paralleli.
- *Esempi di applicazioni:*
 - Applicazioni scientifiche che modellano e simulano fenomeni come le previsioni del tempo,evoluzione del sistema solare etc..
 - Elaborazioni di immagini. La creazione di effetti speciali nei film, etc.
 - Problemi di ottimizzazione di grandi dimensioni