Embedded Systems

Sistemi Operativi per Architetture Parallele

Docente:
Prof. William Fornaciari
Politecnico di Milano
fornacia@elet.polimi.it

Sommario

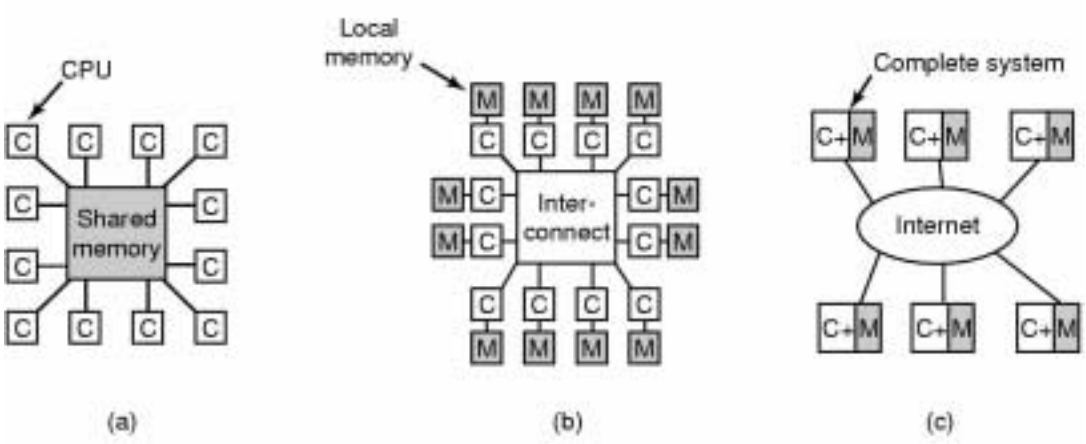
- Introduzione
- Sistemi Multiprocessore
- Sistemi Multicomputer

Introduzione

- Da quando il computer è stato inventato c'è sempre stata richiesta di una potenza di calcolo superiore a quella disponibile
  - Soluzioni
    - Processori con frequenza di clock più elevata
      - Limitazioni fisiche
        - » Dimensioni
        - » Calore
    - Sistemi Multiprocessore
      - Un singolo calcolatore con più processori
        - » Applicazioni tipiche: *Number Crunching*
    - Sistemi Multicomputer
      - Più calcolatori collegati tra di loro e cooperanti
        - » Problema principale: la comunicazione

Introduzione

- Le differenti tecnologie di interconnesione danno origine a differenti tipologie di architetture di sistema
  - Ad esempio:
    - Sistema Multiprocessore a Memoria Condivisa (*a*)
    - Multicomputer a Scambio di Messagi (*b*)
    - Sistema Distribuito (*c*)



Introduzione

- Sistema Multiprocessore a Memoria Condivisa
  - 2..1000 CPU che comunicano tramite memoria condivisa
    - Leggono e scrivono le stesse locazioni di memoria (10..50 ns)
  - Implementazione alquanto complessa
- Sistema Multicomputer a Scambio di Messagi
  - Varie coppie CPU-Memoria Locale collegate tramite una rete ad alta velocità (1..50 μs)
  - Più semplici da costruire ma più difficili da programmare
- Sistema Distribuito
  - Sono sistemi multicomputer collegati tramite una WAN (*Wide Area Network)*
    - *Loosely-coupled vs. Tightly-coupled*
  - I tempi di comunicazione (10..50 ms) impongono una differente utilizzazione di questi sistemi

Sistemi Multiprocessore

- Generalità
- Aspetti HW
- Aspetti SW (SO)

Sistemi Multiprocessore

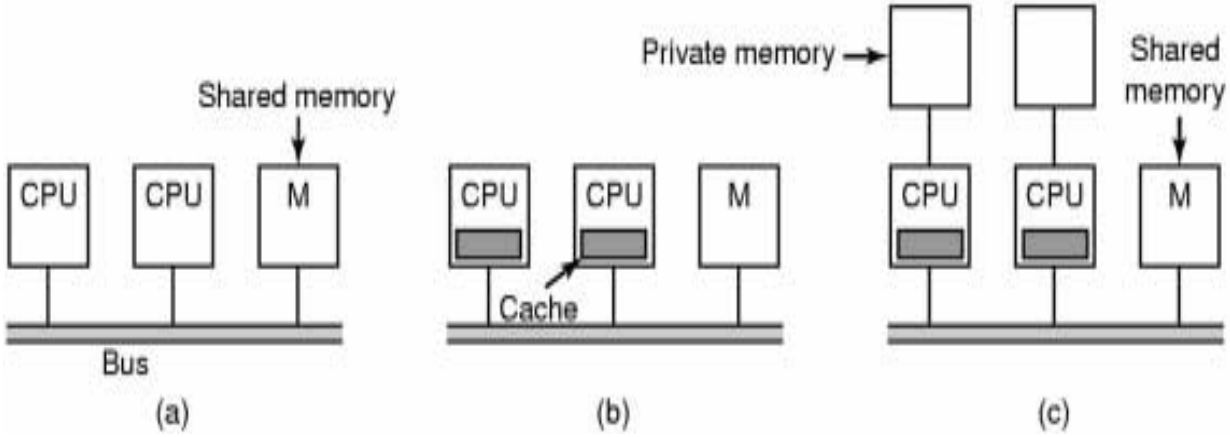
- Sono sistemi in cui 2 o più CPU hanno pieno accesso ad una memoria condivisa
- I programmi vedono lo stesso spazio di indirizzamento (virtuale)
  - Un processore potrebbe fare una STORE e poi una LOAD nella stessa locazione di memoria e trovare un valore diverso perchè un altro processore l'ha modificata
  - Quando organizzata correttamente questa proprietà forma la base della comunicazione inter-processore
- A parte alcuni aspetti (sincronizzazione, *scheduling*) i SO per questi sistemi sono molto simili a quelli classici

Sistemi Multiprocessore

- Aspetti HW
  - Una differenza tra sistemi multiprocessore è legata alla capacità di poter accedere con tempi uniformi a tutte le locazioni di memoria
    - UMA (*Uniform Memory Acces*)
      - *Architetture UMA Simmetric Multi Processor basate su Bus*
      - *Architetture UMA basate su Crossbar Switch*
      - *Architetture UMA basate su Reti di Switch Multistadio*
    - NUMA (*Not Uniform Memory Access*)

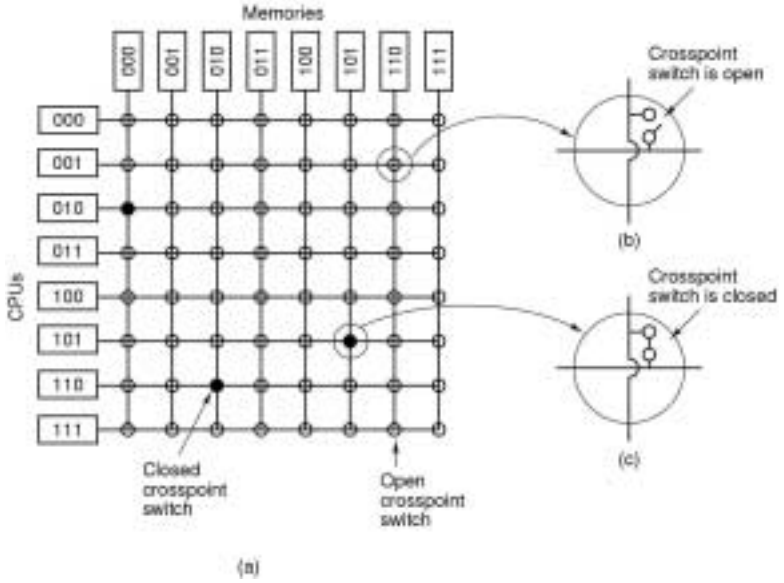
Sistemi Multiprocessore

- Architetture UMA SMP basate su *Bus*
  - Bus singolo senza *cache* (*a*)
    - Contesa per l'accesso al bus: <32 CPU
  - Bus singolo con cache (*b*)
    - Coerenza della *cache*
  - Bus singolo con cahe e memoria privata (*c*)
    - Compilatori appositi



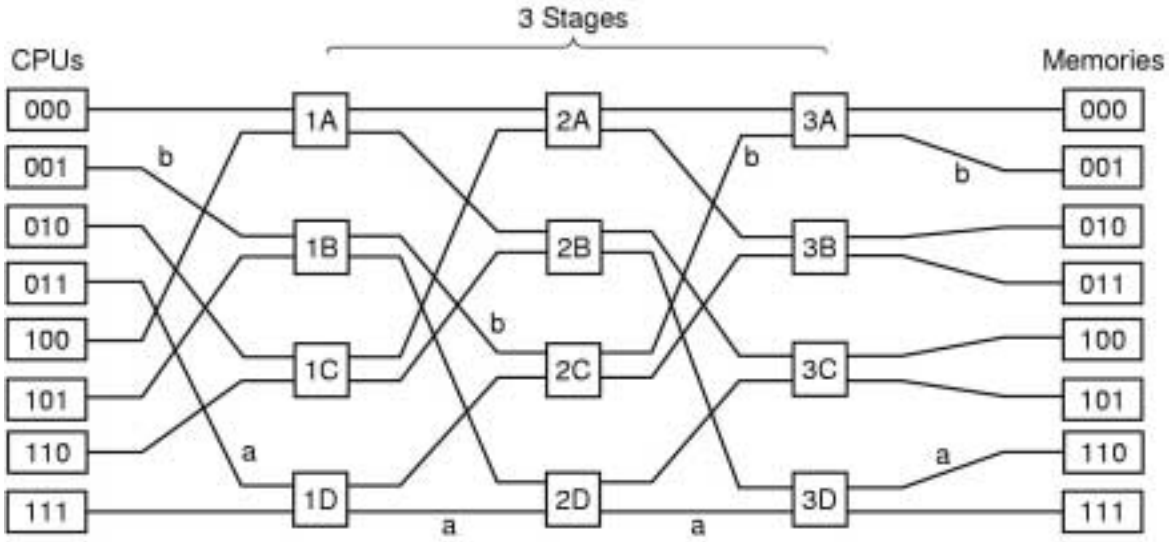
Sistemi Multiprocessore

- Architetture UMA basate su *Crossbar Switch*
  - Permettono di superare le limitazioni imposte dal *bus*
    - Collegamenti tra *n* CPU e *k* memorie (*crosspoint*)
    - Pregio: *Non-Blocking Network*
    - Difetto: il numero di crosspoint cresce come n^2



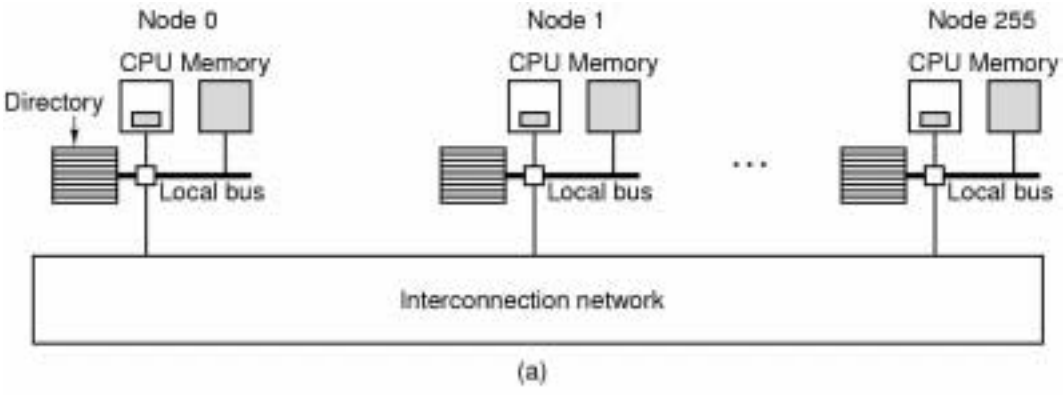
Sistemi Multiprocessore

- Arch. UMA basate su *Reti di Switch Multistadio*
  - *Usano reti costruite tramite* switch *2x2*
    - Pregio: ridotto numero di *switch* (*Omega Network*)
      - Il numero di switch cresce come $(n/2)log_2 n$
    - Difetto: *Blocking Network*
      - E' possibile usare più *switch* per limitare i blocchi



Sistemi Multiprocessore

- Architetture NUMA
  - Indispensabili per poter collegare più di 100 processori
  - Esiste sempre un unico spazio di indirizzamento accessibile con operazioni di LOAD e STORE
  - Gli accessi remoti sono più lenti dei locali
    - Degrado di prestazioni: uso di cache per limitare i tempi



Sistemi Multiprocessore

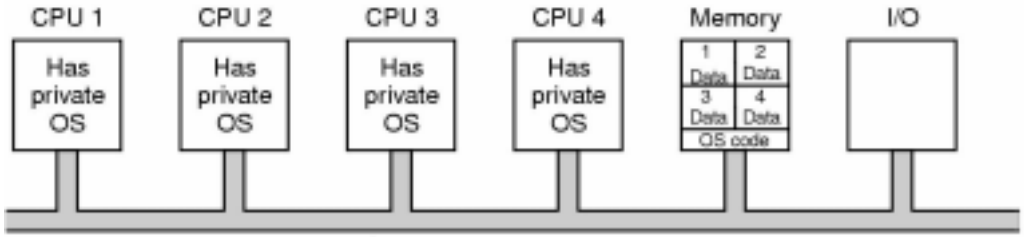
- Aspetti SW (SO)
  - Tipologie di SO per Architetture Multiprocessore
  - La Sincronizzazione nei Sistemi Multiprocessore
  - Lo *Scheduling* nei Sistemi Multiprocessore

Sistemi Multiprocessore

- Tipologie di SO per Architetture Multiprocessore
  - Ogni CPU con il suo SO
  - Sistemi Multirpocessore *Master-Slave*
  - Sistemi Multiprocessore Simmetrici (SMP)

Sistemi Multiprocessore

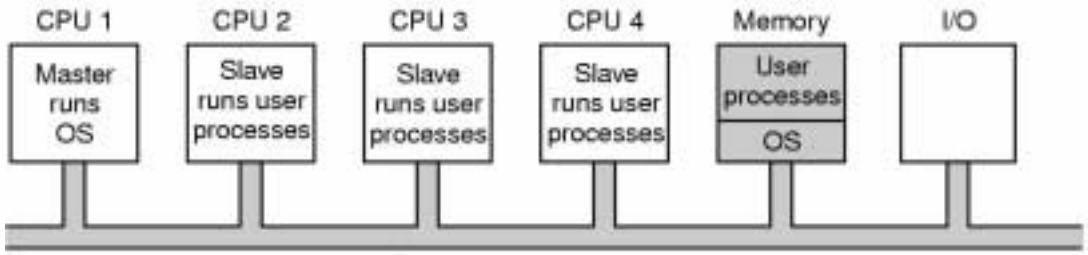
- Tipologie di SO per Architetture Multiprocessore
  - Ogni CPU con il suo SO (replicate solo le strutture dati)



    - Pregi
      - Semplice: porzione di memoria privata ed estendibile
      - Condivisione di risorse (es. Dischi)
      - Comunicazioni inter-processore efficienti
    - Difetti
      - Non c'e' condivisione di processi: carico sbilanciato
      - Non c'e' condivisione di pagine: spreco di memoria e risorse
      - Problemi di consistenza dei buffer per i dispositivi di I/O

Sistemi Multiprocessore

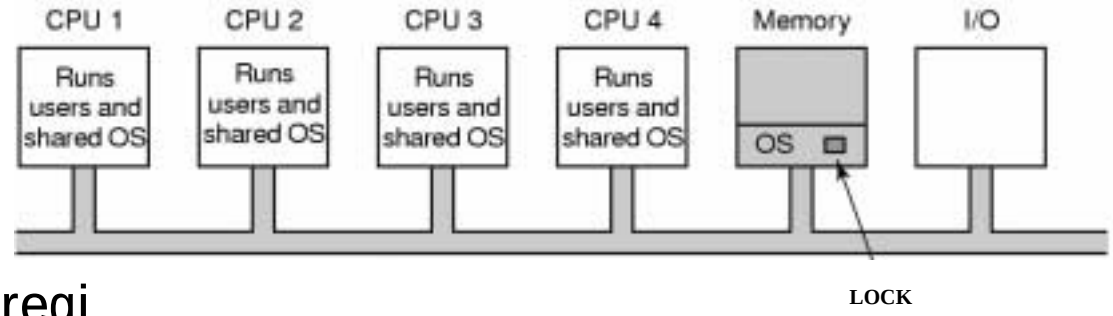
- Tipologie di SO per Architetture Multiprocessore
  - Sistemi Multiprocessore *Master-Slave*



- Pregi
  - Solo la CPU *Master* ha il SO
    - » Raccoglie tutte le chiamate di sistema
    - » Si occupa di smistare i processi: carico bilanciato
  - Condivisione di pagine
  - Una sola copia dei buffer per i dispositivi di I/O
- Difetti
  - Il problema è che al crescere del numero di CPU (>5) il *Master* diventa un collo di bottiglia

Sistemi Multiprocessore

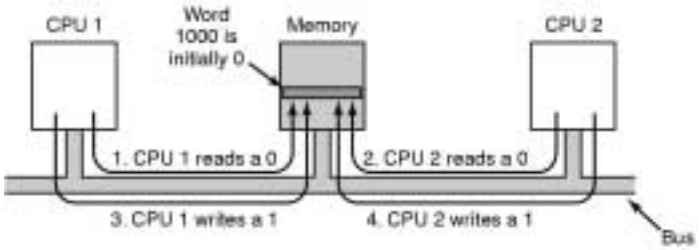
- Tipologie di SO per Architetture Multiprocessore
  - Sistemi Multiprocessore Simmetrici (SMP)



    - Pregi
      - Esiste una sola copia di SO ma ogni CPU lo può eseguire
        - » Ogni CPU esegue le proprie chiamate di sistema
      - Processi e memoria vengono bilanciati dinamicamente
    - Difetti
      - Necessità di eseguire il SO in mutua esclusione: *LOCK*
        - » Accesso al SO: collo di bottiglia!
        - » Dividere il SO in parti indipendenti accessibili in parallelo
        - » E'un'operazione complessa e problematica

Sistemi Multiprocessore

- La Sincronizzazione nei Sistemi Multiprocessore
  - La sincronizzazione tra i processori è fondamentale, soprattutto per quanto riguarda l'accesso esclusivo a risorse critiche
    - Non basta più disabilitare gli *interrupt*
    - Non è più possibile basarsi sulla semplice istruzione TSL



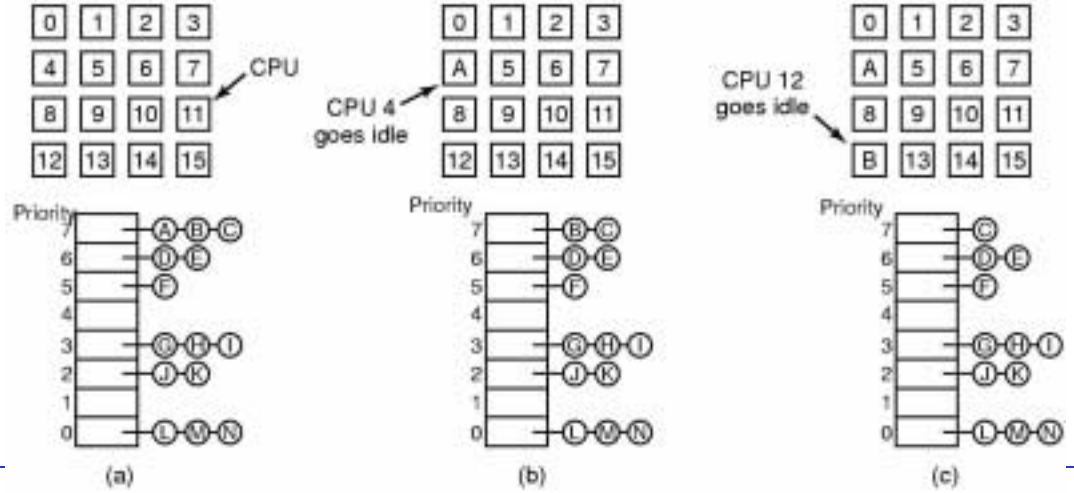
      - La TSL deve poter *lockare* anche il *bus*
        - » Ciò causa spreco di risorse e sovraccarico (*spin lock*)
    - Esistono vari algoritmi per ridurre lo spreco di risorse
      - Tentativi ritardati, liste di attesa, etc.

Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - Lo Scheduling nei Sistemi Multiporocessore è un problema bi-dimensionale: il SO deve decidere quale processo eseguire e su quale CPU eseguirlo
    - *Timesharing*
    - *Space Sharing*
    - *Gang Scheduling*

Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - *Timesharing*
    - Senza considerare le dipendenze tra processi si può utilizzare una singola tabella dei processi per tutto il sistema
    - Ogni processore libero esegue il successivo processo pronto (selezionato in base ad una qualche politica, es. priorità)

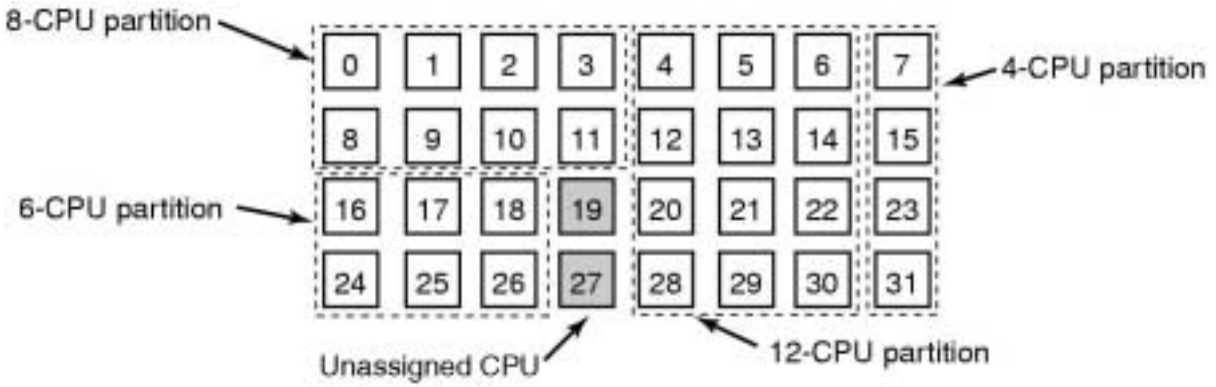


Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - *Timesharing*
    - Pregi: bilanciamento del carico
    - Difetti
      - Contesa per l'accesso alla Tabella dei Processi
      - Poco sfruttamento della *cache* interna al processore
    - Miglioramenti
      - *Affinity Scheduling:* cercare di eseguire un processo sull'ultimo processore che l'ha eseguito
      - *Two-level Algorithm:* un gruppo di processi è assegnato ad una CPU che li gestisce con una struttura dati dedicata
      - Quando una CPU è *idle* prende un processo da qualcun'altra
        - » Bilanciamento del carico
        - » Massimizzazione della *cache affinity*
        - » Riduzione della contesa per la Tabella dei Processi

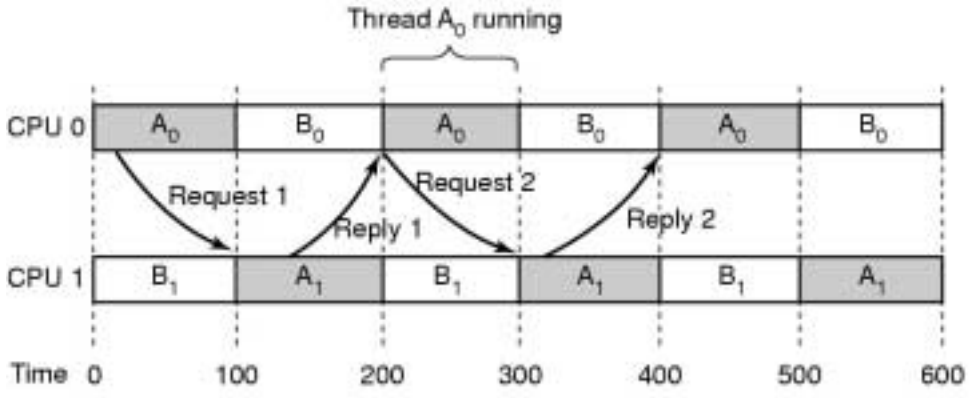
Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - *Space Sharing*
    - Un gruppo di *k* processi (*thread*) correlati viene assegnato a k CPU disponibili
      - In ogni istante l'insieme di CPU è staticamente partizionato in gruppi che eseguono processi tra loro correlati
      - Buono per lavori *batch*: si conoscono le relazioni tra i processi



Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - Lo *Space Sharing* elimina l'*overhead* del cambio di contesto ma le CPU possono rimanere *idle* per molto tempo
  - Non considerando le relazioni tra i processi (*thread*) si possono avere inefficienze dovute alle comunicazioni



  - Alcuni algoritmi cercano di effettuare in contemporanea lo scheduling nel tempo e nello spazio tenendo in considerazione le dipendenze/relazioni tra i processi (*thread*)

Sistemi Multiprocessore

- Lo Scheduling nei Sistemi Multiprocessore
  - *Gang Scheduling:* ha come obbiettivo quello di eseguire in contemporanea i processi (*thread*) correlati
    - Gruppi di processi (*thread*) correlati (*gang*) sono schedulati in modo indivisibile
    - I membri di una *gang* sono eseguiti simultaneamente in *timesharing* da più processori
    - I membri di una *gang* hanno i *time slice* coincidenti
      - » Allo scadere di ogni quanto tutte le CPU sono ri-schedulate

Time slot	CPU 0	1	2	3	4	5
0	A_0	A_1	A_2	A_3	A_4	A_5
1	B_0	B_1	B_2	C_0	C_1	C_2
2	D_0	D_1	D_2	D_3	D_4	E_0
3	E_1	E_2	E_3	E_4	E_5	E_6
4	A_0	A_1	A_2	A_3	A_4	A_5
5	B_0	B_1	B_2	C_0	C_1	C_2
6	D_0	D_1	D_2	D_3	D_4	E_0
7	E_1	E_2	E_3	E_4	E_5	E_6

Sistemi Multicomputer

- Generalità
- Aspetti HW
- Aspetti SW (SO)

Sistemi Multicomputer

- I Sistemi Multiprocessore offrono un semplice modello per la comunicazione ma al crescere del numero di processori sono difficili da costruire e quindi molto costosi

- Sono quindi nati i Sistemi Multicomputer composti da CPU lascamente accoppiate che non condividono memoria
  - *Cluster Computer*
  - *Cluster of Workstation* (COWS)

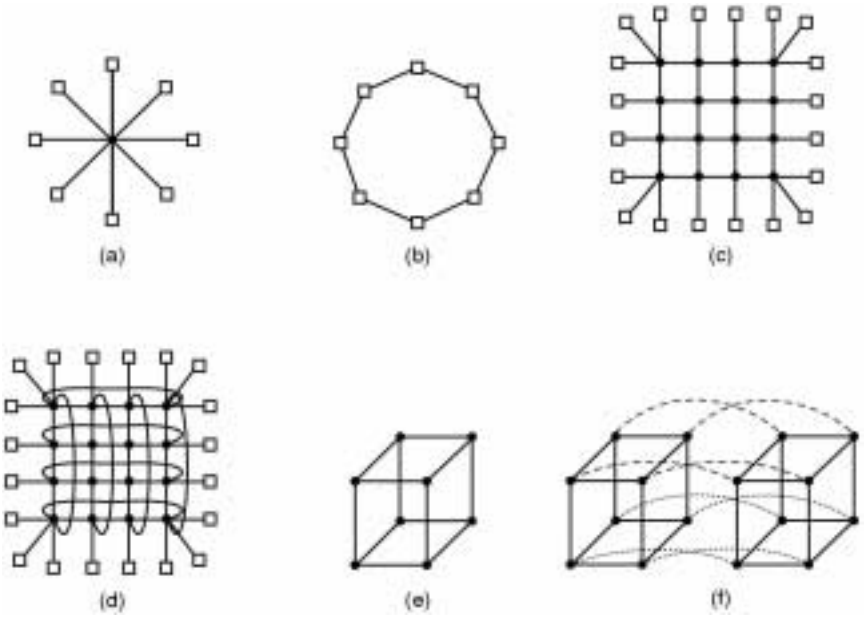
- Si tratta in pratica di normali calcolatori collegati da una rete di interconnessione
  - Il problema questa volta è progettare efficacemente tale rete
  - Il compito è meno arduo rispetto a Sistemi Multiprocessore perchè i tempi in gioco sono di un ordine di grandezza superiore

Sistemi Multicomputer

- Aspetti HW
  - Il nodo base di un Multicomputer consiste in uno o più processori, memoria, un'interfaccia di rete e (alle volte) un *hard disk*
    - Tecnologie di Interconnesisone
    - Interfacce di Rete

Sistemi Multicomputer

- Tecnologie di Interconnesione
  - Ogni nodo è connesso ad altri nodi o a *switch* secondo una determinata topologia
  - Topologie Classiche
    - Stella (*a*)
    - Anello (*b*)
    - *Grid* o *Mesh* (c)
      - Scalabile
      - Diametro = $f(n^{1/2})$
    - Doppio Toro (d)
      - Tollerante ai guasti
    - Cubo (e)
    - Ipercubo (f)
      - Diametro = $f(\log_2 n)$

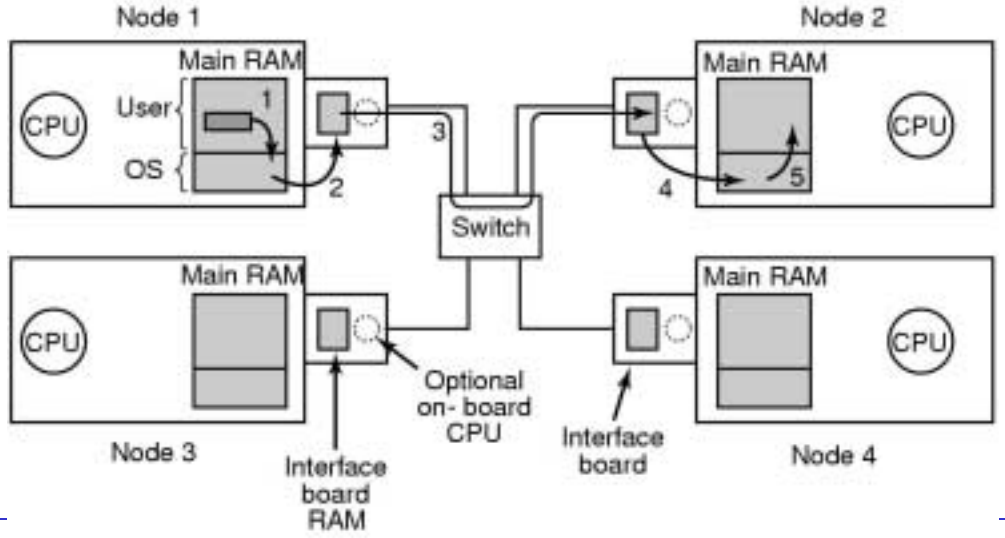


Sistemi Multicomputer

- Tecnologie di Interconnesione
  - Stategie di *Switching*
    - A pacchetto
      - Si trasferisce un pacchetto (per intero) alla volta
      - *Store-and-forward Packet Switching*
        - » Flessibile ed efficiente ma latenza cresce con la dimensione della rete
    - A circuito
      - Percorso predeterminato
      - Non ci sono salvataggi intermedi
        - » Occorre una fase di *setup*
    - *Wormhole Routing*
      - E' una via di mezzo tra le prime due strategie
        - » Un pacchetto è diviso in pezzi più piccoli che fluiscono man mano che il percorso viene stabilito

Sistemi Multicomputer

- Interfacce di Rete
  - Il modo con cui sono costruite e come interagiscono con CPU e RAM influiscono notevolmente sul SO
    - In genere sono dotate di RAM a bordo per mantenere un flusso continuo di bit durante la trasmissione/ricezione dei dati
      - Possono anche avere controller DMA e/o CPU a bordo

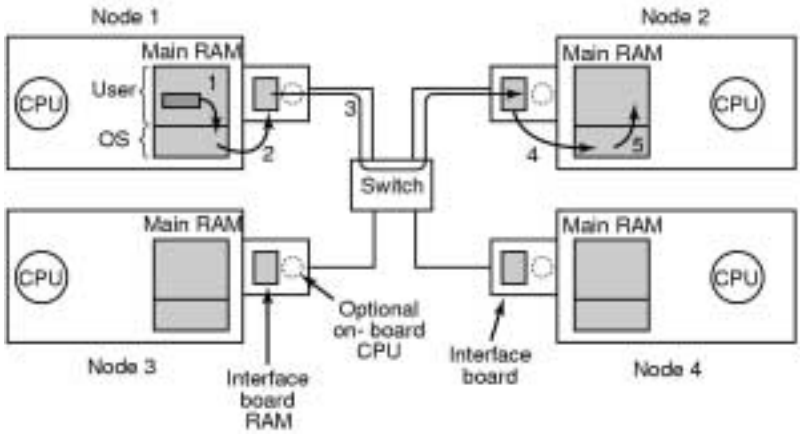


Sistemi Multicomputer

- Aspetti SW (SO)
  - La Comunicazione nei Sistemi Multicomputer
  - Lo *Scheduling* nei Sistemi Multicomputer
  - Bilanciamento del Carico nei Sistemi Multicomputer

Sistemi Multicomputer

- La Comunicazione nei Sistemi Multicomputer
  - Comunicazione di Basso Livello
    - Copia dei pacchetti



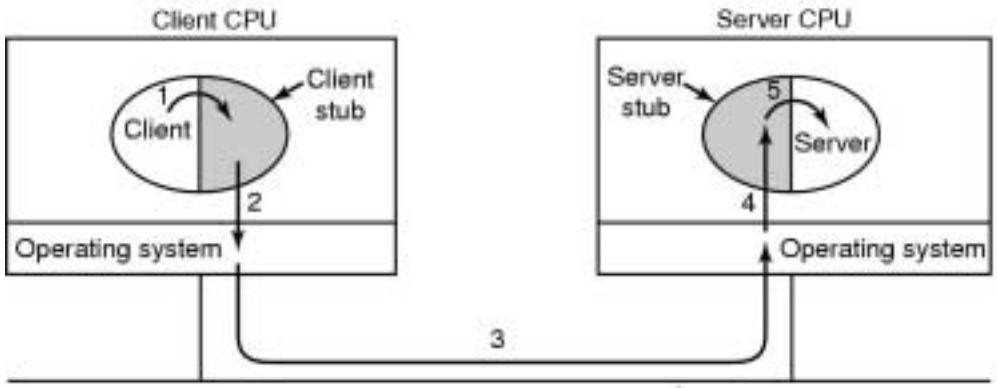
    - Minimizzare le copie dei pacchetti: interfaccia di rete mappata in *user-space*
      - Problemi di condivisione tra processi
      - Problemi di accesso da parte del kernel
        - » Doppia interfaccia di rete

Sistemi Multicomputer

- La Comunicazione nei Sistemi Multicomputer
  - Comunicazione di Livello Utente
    - I processi si scambiano messaggi tramite opportune chiamate di sistema
      - *Send* and *Receive*
        - » In questo modo la comunicazione è esplicitamente gestita dall'utente
    - Queste comunicazioni possono essere blocccanti (sincrone) o non bloccanti (asincrone)
      - Nel secondo caso l'elaborazione può continuare a patto di non usare il buffer contenente il messaggio spedito/ricevuto fino a trasferimento completato
      - Servono dei meccanismi per avvisare il mittente che il buffer è utilizzabile
        - » Es. *interrupt*, *pop-up thread*

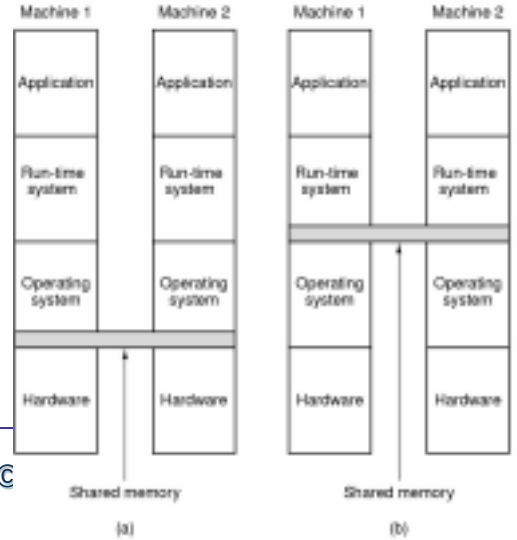
Sistemi Multicomputer

- La Comunicazione nei Sistemi Multicomputer
  - Remote Procedure Call (RPC)
    - Offrono un paradigma diverso da quello basato sull'I/O
      - Chiamate a procedure rersidenti su un altro calcolatore
        - » Normale passaggio di parametri
        - » I processi utente fanno chiamate a procedura locali al *client stub* (che gira in spazio utente)
        - » Tali procedure hanno lo stesso nome di quelle *server* e si occupano del vero I/O



Sistemi Multicomputer

- La Comunicazione nei Sistemi Multicomputer
  - Distributed Shared Memory (DSM)
    - Permette di mantenere il concetto di memoria condivisa
      - Con la DSM le pagine sono dislocate nelle varie memorie locali
      - Quando una CPU effettua una *load* (*store*) su una pagina che non ha, avviene una chiamata al sistema operativo che provvede a recuperarla facendosela spedire appena possibile
        - » *Page fault* remoto
    - Differenze con la vera Shared Memory
      - Sistemi Multiprocessore (*a*)
        - » Gestione HW
      - Sistemi Multicomputer (*b*)
        - » Gestione SW (SO)



Sistemi Multicomputer

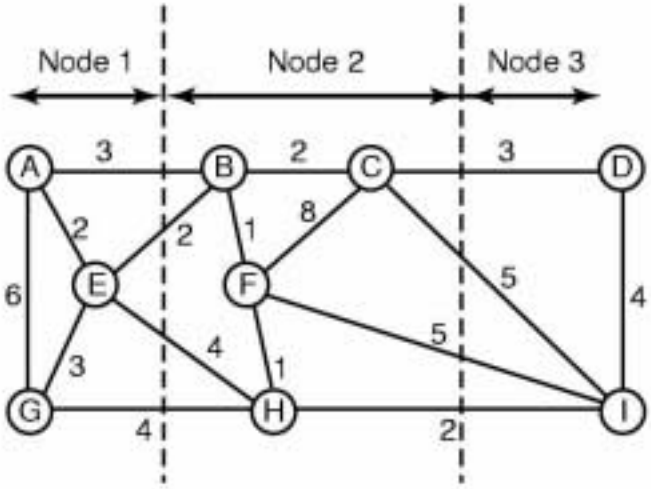
- Lo *Scheduling* nei Sistemi Multicomputer
  - Nei Sistemi Multiprocessore tutti i processi risiedono nella stessa memoria e potenzialmente ogni CPU può eseguirne uno qualunque
  - Nei Sistemi Multicomputer ogni CPU ha un certo insieme di processi da eseguire e non è fattibile (per l'elevato costo della comunicazione) uno scambio dinamico di questi
  - In pratica lo scheduling è locale e quindi quello classico
  - Diventa però fondamentale l'allocazione dei processi sui vari nodi ai fini del bilanciamento del carico e della minimizzazione delle comunicazioni inter-nodo

Sistemi Multicomputer

- Bilanciamento del Carico nei Sistemi Multicomputer
  - E' di estrema importanza proprio perchè lo scheduling locale non permette di intervenire a posteriori
    - Fondamentale differenza con i Sistemi Multiprocessore
  - *Processor Allocation Algorithm*
    - *Graph-Theoretic Deterministic Algorithm*
    - *Sender-Initiated Distributed Heuristic Algorithm*
    - *Receiver-Initiated Distributed Heuristic Algorithm*

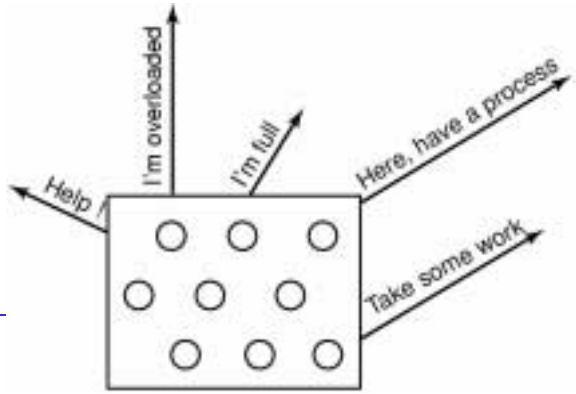
Sistemi Multicomputer

- Bilanciamento del Carico nei Sistemi Multicomputer
  - *Graph-Theoretic Deterministic Algorithm*
    - Si basano su stime dei requisiti di CPU e memoria da parte dei processi e di traffico medio sulla rete
    - Cercano l'allocazione che minimizza il traffico sulla rete
      - Teoria dei grafi: insiemi di taglio tali da soddisfare dei vincoli (CPU e memoria) e minimizzarne altri



Sistemi Multicomputer

- Bilanciamento del Carico nei Sistemi Multicomputer
  - *Sender-Initiated Distributed Heuristic Algorithm*
    - Quando un processo viene creato esso è eseguito localmente a meno che il nodo in questione non sia sovraccarico
      - Il carico è calcolato con opportune metriche
    - Se il nodo è sovraccarico questo contatta altri nodi (a caso) e se ne trova uno con il carico più basso del suo gli spedisce il nuovo processo
      - Dopo *k* tentativi vani il processo è eseguito localmente



Sistemi Multicomputer

- Bilanciamento del Carico nei Sistemi Multicomputer
  - *Receiver-Initiated Distributed Heuristic Algorithm*
    - E' duale al precedente: quando un processo termina e il carico e basso, il nodo contatta (a caso) altri nodi in cerca di processi da eseguire
      - Dopo *k* tentativi smette di chiedere
    - In questo modo sistemi già carichi non sono costretti a fare lavoro in più per cercare collaboratori
      - Si crea motlo traffico quando il sistema è molto scarico
  - Si possono combinare i due algoritmi
    - Offerte e richieste simultanee
    - Alternativa migliore alla ricerca casuale
      - Elenco dei nodi spesso sovraccarichi
      - Elenco dei nodi spesso liberi

