Architettura e programmazione dei sistemi multiprocessore

di Giuseppe Cardinale Ciccotti

parte seconda

La principale caratteristica di un sistema multiprocessore è la possibilità per ciascun processore, di condividere un insieme di moduli di memoria e, se possibile, di dispositivi di I/O. La condivisione delle risorse è quindi affidata ad una rete di interconnessione. Questa può essere divisa in due blocchi logici dei quali il primo è responsabile delle comunicazioni tra processori e moduli di memoria e l'altro tra i processori e i dispositivi di I/O. Ci sono molte forme diverse per realizzare tali reti di interconnessione; presenteremo quattro schemi di organizzazione, che rispondono alle più comuni realizzazioni

Bus comuni o condivisi temporalmente (time shared)

Il più semplice sistema di interconnessione tra processori multipli è un canale di comunicazione comune che connette tutte le unità funzionali. Un esempio di tale schema è visualizzato in figura 1. Questo canale di comunicazione è spesso chiamato «bus time shared o bus comune»; la dizione «time shared» significa «condiviso temporalmente». Il bus è in generale un'unità passiva che non ha nessuna componente attiva. Le operazioni di trasferimento sono completamente controllate dalle interfacce di bus delle unità trasmittenti e riceventi. Tuttavia in un sistema multiprocessore, il bus è una risorsa condivisa e deve essere previsto un meccanismo atto a risolvere le contese per il possesso del bus stesso.

I metodi per la risoluzione dei conflitti sono basati sui concetti di priorità statica o dinamica, di code servite con disciplina primo arrivato primo servito (first-in first-out) e di daisy chain. Bisogna tener presente che in un sistema multiprocessore, non esiste più il concetto di unità master e unità slave; infatti ogni unità può comportarsi da master cioè può iniziare a gestire una trasmissione e

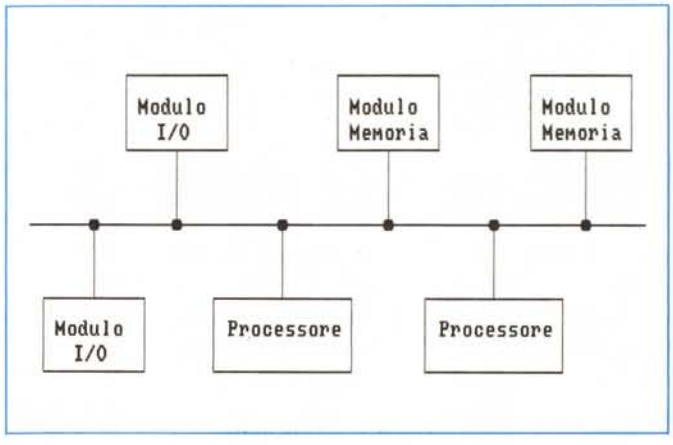
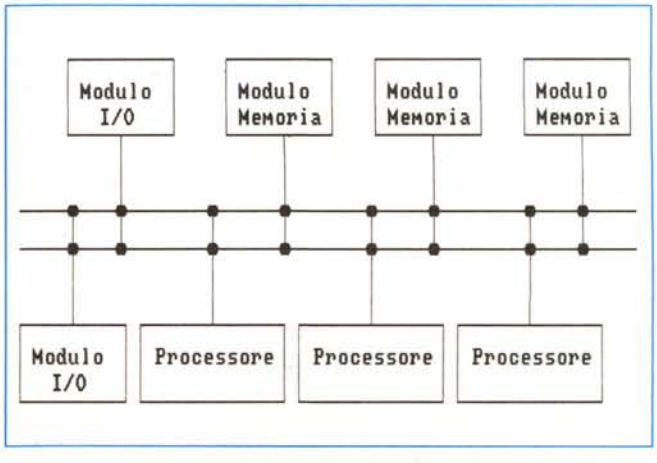


Figura 1 Multiprocessore con organizzazione a bus singolo condiviso.

Figura 2 Organizzazione a doppio bus.



di conseguenza prendere il possesso del bus. Risulta perciò chiaro che ogni unità funzionale del sistema debba «affacciarsi» sul bus comune tramite un'interfaccia che possa comportarsi sia da master che da slave. In generale, una unità che voglia iniziare un trasferimento deve eseguire i seguenti passi:
1 - verificare la disponibilità del bus.
2 - Se il bus è disponibile, verificare la disponibilità dell'unità ricevente.
3 - Se il punto 2 è verificato, specificare un'eventuale operazione sui dati.
4 - Iniziare la trasmissione.

L'unità ricevente deve, per parte sua, essere in grado di riconoscere se è presente sul bus un messaggio indirizzatole e rispondere al mittente in modo opportuno.

L'organizzazione a singolo bus condiviso è quindi semplice e soprattutto economica, tuttavia presenta due inconvenienti piuttosto pesanti. Il primo è un problema di tolleranza ai guasti che è evidentemente assai basso, infatti un malfunzionamento di una sola unità potrebbe bloccare il sistema, per esempio non rilasciando più il bus, bisogna perciò prevedere, come minimo, delle unità indipendenti che d'autorità tolgano il possesso del bus e ripristinino il funzionamento corretto del sistema; queste unità sono chiamate, in maniera appropriata, «watch-dog» cioè cane da guardia. Il secondo è un problema di espandibilità del sistema, in quanto all'aumentare delle unità di processo o di memoria sul bus, vi è un incremento di conflitti per l'accesso al bus con un inevitabile degradamento dell'efficienza del sistema e un aumento della complessità della logica di arbitraggio. Rifacendoci alle considerazioni fatte riguardo al modello di Von Neumann nei primi articoli di questa serie sui calcolatori paralleli, le prestazioni totali del sistema sono determinati dalle caratteristiche del canale che collega il PE alla memoria. Nel caso di sistemi a bus condiviso, il canale è costituito dal bus stesso e perciò, per aumentare le prestazioni del sistema, spesso si prevedono moduli di memoria e di I/O privati, il sistema nel suo complesso quindi ha un numero maggiore di canali.

Per alleviare le limitazioni di questa architettura ad un solo bus, si può adottare quella mostrata in figura 2 con due bus bidirezionali, questa soluzione ha il pregio di aumentare le prestazioni tuttavia pesa sull'organizzazione del sistema in quanto c'è la necessità di stabilire volta per volta a quale bus è assegnata la trasmissione; in questo caso i due bus diventano necessariamente un'uni-

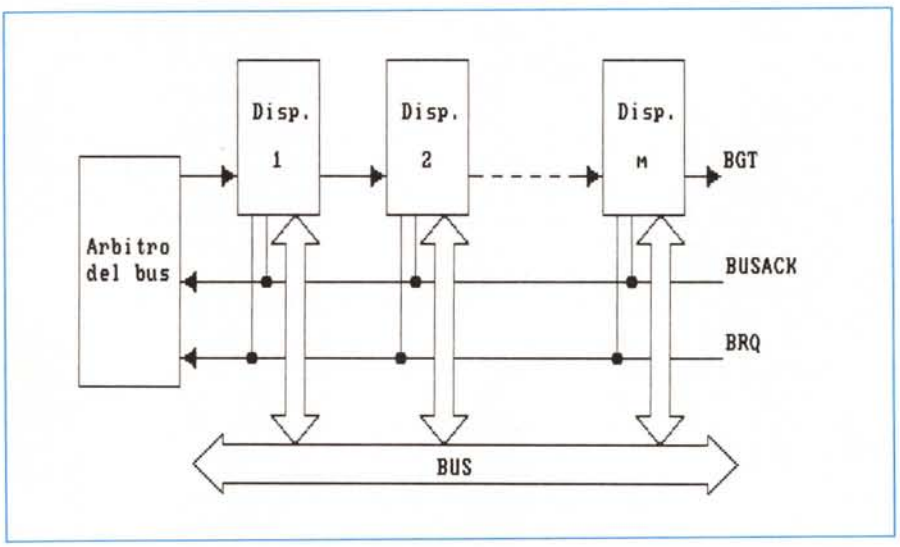


Figura 3 - Bus condiviso arbitrato con algoritmo «Daisy Chain statica».

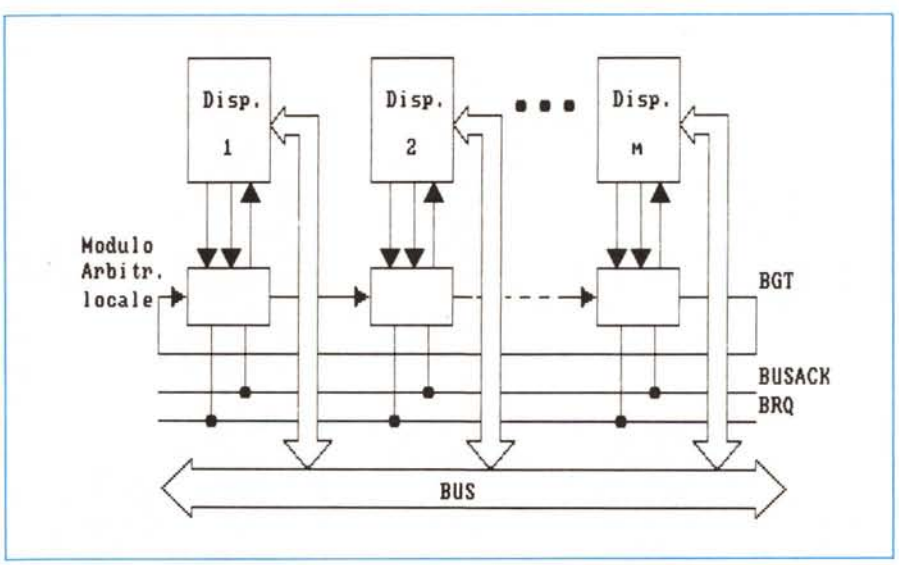


Figura 4 - Bus condiviso arbitrato con algoritmo «Daisy Chain rotante».

tà attiva. Molti sono i fattori limitanti le prestazioni del bus, questi includono il numero di dispositivi attivi sul bus, l'algoritmo di arbitraggio del bus la centralità del controllo, il numero di bit del dato trasferito, la sincronizzazione delle trasmissioni e il riconoscimento di eventuali errori.

Di seguito esamineremo un certo numero di algoritmi di arbitraggio che controllano l'accesso al bus dei dispositivi in conflitto. Le realizzazioni commerciali sono basate su algoritmi relativamente semplici, al fine di permetterne l'implementazione hardware e assicurarne un'elevata velocità di esecuzione.

Algoritmi a priorità statica

Nello schema a priorità statica, quando due o più dispositivi richiedono l'uso del bus, questo è assegnato a quello con priorità maggiore. La priorità può essere implementata in varie maniere, la più semplice e diffusa è quella detta «daisy chaining», nella quale la priorità è stabilita dall'ordine del dispositivo lungo la linea di assegnazione (Grant). Il servizio perciò è assegnato al dispositivo più vicino all'arbitro tra quelli che hanno richiesto la risorsa.

In figura 3 si possono notare le linee di richiesta del bus, BRQ, comune a tutti i dispositivi e di conferma, BUSACK, anch'essa comune. Quando BRQ è asserito, l'arbitro del bus attiva BGT se BUSACK indica che il bus è libero. BGT giunge al primo dispositivo nella catena; se tale dispositivo aveva asserito BRQ, blocca la propagazione di BGT e prende il controllo del bus settando BUSACK. Non appena ha finito il trasferimento, il dispositivo libera il bus resettando BUSACK; se un altro dispositivo richiede il bus, BRQ rimane asserito e perciò l'arbitro asserisce di nuovo BGT. Questa volta il dispositivo a priorità più alta che è stato già servito lascerà passare il Grant che si propaga fino al successivo dispositivo ripetendo la stessa procedura di handshaking. Con questa disciplina di arbitraggio i dispositivi fisicamente più vicini al bus sono ovviamente favoriti e nel caso peggiore i dispositivi a priorità più bassa potrebbero non essere mai serviti; in un sistema del genere è perciò buona norma, a meno di particolari esigenze, assegnare priorità più alte a dispositivi che occupano per un tempo minore il bus. Un'altra pecca di questo sistema è che il tempo richiesto per determinare quale dispositivo ha priorità maggiore non è fisso; nei sistemi Digital VAX 11/780 è stato implementato uno schema chiamato «risoluzione parallela delle priorità» per ovviare a tale inconveniente.

Algoritmi a partizioni di tempo fisse

Un altro algoritmo di arbitraggio molto comune consiste nell'assegnare il bus ad ogni dispositivo del sistema per un tempo prefissato. In questo modo il tempo in cui il bus è disponibile per le comunicazioni, viene diviso in tante «fette» temporali offerte a ciascun dispositivo a rotazione, secondo uno schema detto round-robin. Se un dispositivo non ha necessità del bus nel momento in cui gli spetta il controllo del bus, nessun altro dispositivo può prendere il possesso del bus che perciò rimane inutilizzato. Questa disciplina di arbitraggio usualmente indicata in letteratura come «fixed time slicing», è spesso adottata nel caso di bus sincrono nel quale tutti i dispositivi sono sincronizzati su un clock comune. Lo schema a partizioni di tempo fisse è intrinsecamente simmetrico, infatti il servizio assegnato a ciascun dispositivo non dipende né dalla sua posizione sul bus né da alcun'altra caratteristica logica o fisica del dispositivo o del bus. In particolare se ci sono m dispositivi, ognuno ha il controllo del bus esattamente per un tempo fissato pari a una partizione ogni m. Questa soluzione ha il vantaggio di fissare un tempo massimo di attesa per un servizio richiesto da un dispositivo, tuttavia le prestazioni migliori si ottengono nel caso di massimo carico cioè quando tutti i dispositivi hanno necessità del bus; solo in tale condizione il bus è totalmente utilizzato. Bisogna notare comunque che un'espansione del sistema, con conseguente aumento dei dispositivi sul bus, comporta un aumento dei tempi di attesa poiché bisogna aumentare il numero delle partizioni, e se anche si diminuisce il tempo di ognuna di queste, per completare una comunicazione saranno necessarie più partizioni. Tuttavia la semplicità e le caratteristiche di simmetria spiegano la diffusione che tale schema ha avuto.

Algoritmo a priorità dinamica

I due algoritmi precedenti hanno il difetto di non adattarsi alle varie condizioni di carico in cui il sistema si trova: è lampante l'inefficienza dell'algoritmo a partizioni di tempo fisse nel caso che uno solo di m dispositivi utilizzi il bus. In questo caso tale risorsa verrà utilizzata solo per 1/m del tempo di ciclo di bus. Per evitare tali situazioni, si può modificare la logica di assegnazione delle priorità in modo che il controllo sia ceduto ad un dispositivo o ad un altro in modo da massimizzare l'utilizzo del bus. L'algoritmo può essere strutturato in modo da non favorire alcun dispositivo, oppure può privilegiare determinati dispositivi per esempio assegnando loro soltanto priorità alte. I due concetti fondamentali per la permutazione dinamica delle priorità sono quello detto del «daisy chain rotante», DCR, e quello del «dispositivo da più tempo in attesa». Quest'ultimo consiste nell'assegnare la priorità maggiore a quello che, tra i dispositivi richiedenti, da più tempo attende sia esaudita la sua richiesta. Questo effetto

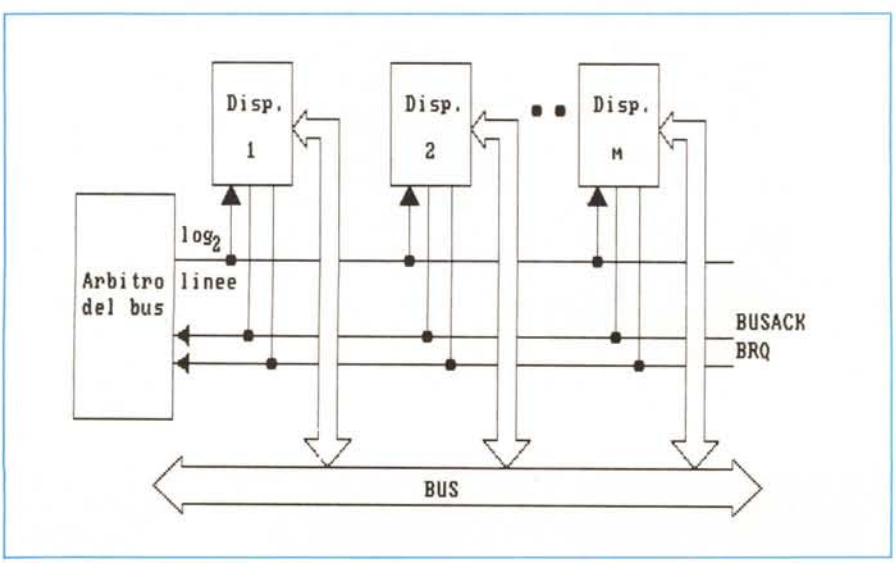


Figura 5 - Bus condiviso arbitrato con algoritmo «FIFO» implementato mediante $\log_2$ linee di polling.

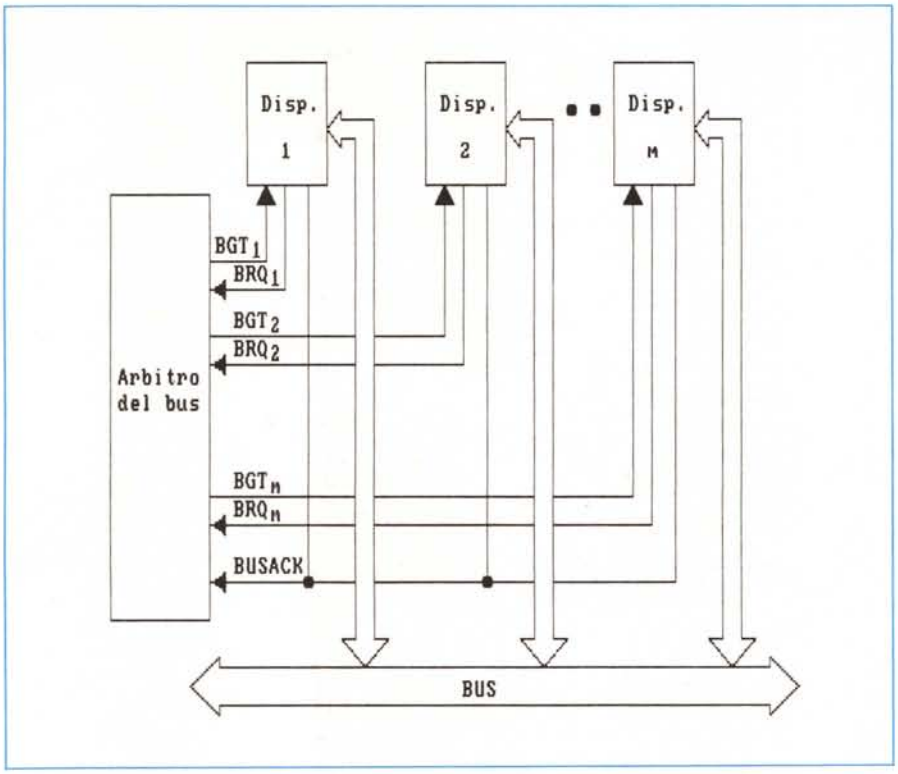


Figura 6 - Bus condiviso arbitrato con algoritmo «FIFO» implementato mediante linee di richiesta indipendenti.

può essere ottenuto riassegnando le priorità dopo ogni ciclo di bus.

Nello schema DCR a differenza dello schema daisy chain a priorità fissato, non esiste un controllo centralizzato e la linea di bus-grant è portata dall'ultimo dispositivo indietro al primo in un anello chiuso, come mostrato in figura 4. L'ultimo dispositivo che ha avuto accesso al bus, viene utilizzato come arbitraggio per la successiva assegnazione (il primo accesso è naturamente arbitrario). La priorità di ciascun dispositivo per ogni arbitraggio è determinata dalla distanza (in una determinata direzione) sulla linea di bus-grant dal dispositivo che in quel momento agisce come arbitro; in figura 4 il dispositivo precedente è quello a priorità minore.

Algoritmo primo arrivato, primo servito

In questo schema, più conosciuto probabilmente come First In First Out (FIFO), una richiesta è semplicemente servita nell'ordine stesso in cui è ricevuta. Questo schema è simmetrico perché nessun dispositivo è favorito per l'accesso al bus. È stato dimostrato che, se il tempo di trasferimento sul bus è fissato, lo schema FIFO offre il minor tempo di attesa medio fra tutti i possibili schemi. Tuttavia presenta due difficoltà implementative; una è che l'hardware di arbitraggio deve predisporre un meccanismo per registrare l'ordine di arrivo di tutte le richieste pendenti, a differenza degli altri algoritmi analizzati. La seconda, più importante, riguarda l'impossibilità di definire un intervallo di tempo tanto piccolo tale da poter isolare fisicamente la singola richiesta senza collisioni con altre. In questa situazione è impossibile distinguere correttamente l'ordine di arrivo delle richieste. Quindi da un punto di vista «filosofico», qualsiasi implementazione di uno schema FIFO è un'approssimazione.

Per realizzare tale algoritmo può essere adoperata la tecnica del «polling» oppure di attivare linee di richiesta indipendenti. In figura 5, viene illustrato lo schema di un arbitro di bus che usa il polling; la linea di bus-grant è sostituita da $\log_2$ (m) linee di polling, ognuna connessa ad un dispositivo. In risposta ad una richiesta di accesso al bus, segnalata da BRQ, l'arbitro interroga i singoli dispositivi immettendo in sequenza i codici binari da 1 a m sulle linee di polling. Quando un dispositivo richiedente, Di, riconosce il proprio indirizzo, asserisce BUSACK per indicare che il bus è occupato. L'arbitro allora termina il processo di polling e assegna il controllo del bus a Di. L'accesso è mantenuto finché il dispositivo asserisce BUSACK. Notate che la priorità di un dispositivo è determinata dalla posizione del suo codice nella sequenza di polling.

In figura 6 vedete invece uno schema in cui per ogni dispositivo sono predisposte le linee di BRQ e BGT. Questa tecnica di richiesta può permettere l'implementazione di molti degli algoritmi presentati.

Per incrementare le prestazioni è necessario, come ormai sappiamo, aumentare il numero di canali tra processori e memoria; seguendo tale direttiva la rete di interconnessione nei sistemi multiprocessore può assumere le stesse caratteristiche già viste negli Array Processor. Avremo perciò strutture crossbar e reti di interconnessione statiche o dinamiche che non presentano problemi diversi da quelli già considerati se non per il fatto che i processi nei sistemi multiprocessore sono inerentemente asincroni e influenzano la gestione delle comunicazioni.

Organizzazione di memorie parallele

In un sistema multiprocessore l'organizzazione della memoria assume grande importanza, poiché l'esecuzione asincrona dei processi non consente di prevedere a priori i tempi in cui si accede alla memoria. L'effetto diretto è che bisogna organizzare la memoria in modo efficiente rispetto alla complessità delle connessioni e dei tempi di accesso, tuttavia basandosi su considerazioni necessariamente generali. Un sistema è quello di predisporre un certo numero di moduli di memoria doppia porta connessi ai processori come in figura 7. Ogni processore vede un modulo di memoria come memoria privata ma tramite il bus può accedere anche agli altri moduli. La memoria privata serve a mantenere i dati dei processi attivi sul processore rispettivo; questa memoria viene detta «Home memory» e non deve contenere informazioni relative a processi attivi su altri processori. In questo modo è ovvio che non si generino conflitti di memoria. Tuttavia la natura stessa del multiprocessing comporta la comunicazione dei dati, e quindi saranno necessari alcuni moduli devoluti al compito di memoria comune. I processori potranno accedere ai dati contenuti in tali moduli tramite una rete di interconnessione qualsiasi, come un bus o dei crossbar oppure delle reti riconfigurabili secondo gli schemi visti per gli Array Processor; in generale questa rete sarà più lenta del canale privato verso la Home memory. Può essere necessario perciò predisporre delle «cache memory» che permettano una gestione più veloce dell'accesso ai dati condivisi. Le cache memory sono speciali memorie la cui caratteristica

principale è l'elevata velocità di accesso, tuttavia la tecnologia che permette tali velocità non consente un'integrazione comparabile con le memorie dinamiche; usualmente la grandezza di una cache memory è di qualche decina di Kbyte. Lo scopo di disporre tali memorie è quello di conservare in esse i dati che vengono più frequentemente richiesti dal processore; ad ogni richiesta la cache memory controlla il contenuto delle proprie locazioni e soltanto se il dato non è presente passa la richiesta alla memoria esterna, in modo invisibile al processore.

Ultimamente sono stati prodotti dispositivi che integrano cache memory sempre più grandi, questo è dovuto all'incremento del clock dei processori a cui non è corrisposto un pari incremento delle memorie dinamiche. Frequenze di clock intorno a 50 MHz hanno quindi costretto i progettisti a integrare le cache memory nei processori stessi per sostenere tali velocità. Il largo uso di queste memorie ha necessariamente affinato le loro modalità di gestione, in particolare la soluzione del problema della frequenza dell'aggiornamento dei dati nella cache memory.

La coerenza dei dati nei sistemi multicache

In un sistema multiprocessore con una cache per processore, si pone un ulteriore delicato problema: quello di garantire la coerenza dei dati nelle diverse cache del sistema. Può accadere infatti che copie di uno stesso dato siano presenti in diverse cache memory; poiché i processi non hanno espliciti meccanismi di sincronizzazione fra di essi, possono verificarsi gravi situazioni di inconsistenza dei dati. Per esempio, poniamo il caso di un processo A, attivo sul processore i, che produce un dato x, utilizzato da un processo B, eseguito dal processore j diverso da i. Il processo A scrive un nuovo x nella sua cache mentre B usa il vecchio valore di x contenuto nella sua cache memory perché quest'ultima non è stata aggiornata; ricordate che le cache memory sono trasparenti al processore e quindi anche ai processi. Il processo B continuerà ad usare il vecchio x finché il nuovo valore di x sarà copiato nella sua cache memory. Vale a dire che avremo risultati scorretti pur essendo il programma e l'algoritmo corretti sintatticamente e semanticamente.

Un'altra situazione d'errore può verificarsi in ambiente multiprogrammato; supponiamo che il processo in esecuzione su un processore sia interrotto e messo in stato di attesa da un interrupt o da un qualsiasi altro evento. Successivamente il processo può essere assegnato dall'algoritmo di scheduling ad un altro processore, i dati utilizzati con maggior frequenza del processo non saranno ovviamente presenti nella cache del nuovo processore e saranno perciò letti dalla memoria comune. Questi dati non sono corretti perché l'ultimo aggiornamento è nella cache memory del processore precedente! Il processo è eseguito come si dice «fuori dal contesto». Come si può intuire tali errori sono assai difficili da scoprire, è quindi indispensabile garantire la consistenza dei dati nelle cache aggiornandoli con qualche tecnica.

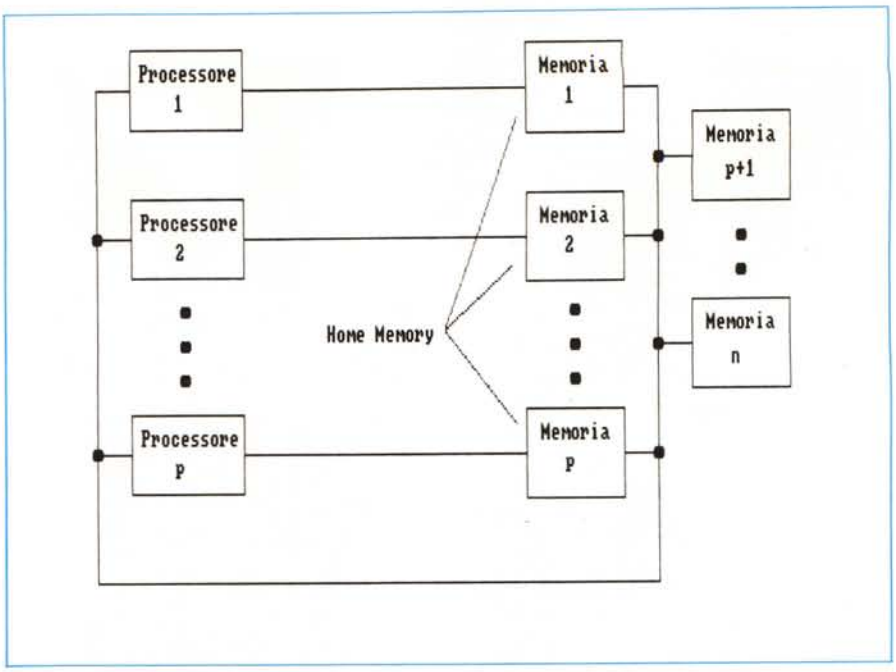


Figura 7 - Schema concettuale di sistema con Home Memory.

Due sono le metodologie usate per raggiungere questo scopo: la prima viene indicata col nome di «controllo statico della coerenza» e si basa sull'idea di distinguere i dati in «cacheabili» e «non-cacheabili». Al primo tipo apparterranno i dati privati dei task di un processo e al secondo quelli comuni, associando un indicatore, un «tag», a ciascun dato la cache memory può riconoscere se conservare o no il dato. Il problema dell'applicazione del tag può essere risolto in fase di compilazione del programma. Se si usa un linguaggio strutturato, i task sono gestiti dal programmatore e il programma è eseguito in ambiente uniprogrammato, il compilatore può essere in grado di applicare i tag opportunamente, in modo automatico. La maggior parte dei compilatori commerciali non è tuttavia in grado di automatizzare il «tagging» e il programmatore deve indicare espressamente se il dato è privato o comune a tutti i task.

Il «controllo dinamico della coerenza» è l'altra metodologia proposta per risolvere il problema della coerenza nelle cache memory, risulta più flessibile ed efficiente ma è sicuramente più costoso e complesso. In questo schema sono permesse copie multiple tra i dati, ma non appena un processore modifica una locazione x nella propria cache, deve controllare tutte le altre cache per invalidare le eventuali copie. Questa operazione è detta «Interrogazione incrociata». L'implementazione più semplice di questo schema prevede che tutte le cache memory siano collegate da un bus ad alta velocità attraverso il quale il processore che aggiorna un dato nella cache, trasmette l'indirizzo di memoria del dato modificato. Questo segnale fa sì che gli altri processori rileggano il dato in questione dalla memoria centrale. Naturalmente è necessaria una logica di handshaking per assicurarsi che la lettura avvenga dopo che il processore abbia scritto il dato nella memoria comune. È ovvio che un'operazione di interrogazione incrociata invalida un dato solo se questo è presente nella cache interrogata.

Conclusioni

Proseguendo nel nostro «viaggio» nel mondo della elaborazione parallela, abbiamo analizzato due aspetti particolari ma importanti dei sistemi multiprocessore. La rete di interconnessione e l'organizzazione della memoria riguardano più la struttura del sistema che il suo schema logico; la prossima volta, invece, dedicheremo la nostra attenzione ai sistemi operativi per multiprocessore.

MC

AT 286 12MHz

80286 16MHz operativi, 1Mb RAM esp. a 4Mb su piastre M/B SUNTAC gestore EMS, controller 2FD 2HD, Floppy 1,2Mb o 1,44Mb, HD 20 Seagate veloce formato 3,1/2, Tastiera Italiana 101 tasti, Scheda video Duale, Monitor 14" monocrom. basculante bifreq., 1 parallela , 2 seriali

TUTTO A LIRE 1.650.000

AT 286 16MHz

80286 21MHz operativi, 1Mb RAM esp. a 8Mb su Piastra M/B NEAT-EMS gestore EMS Shadow RAM per Bios, controller per 2FD e 2HD, Floppy da 1,2Mb o 1,44Mb, HD da 20Mb Seagate veloce 3,1/2, Tastiera Italiana 101 tasti, scheda video Duale, Monitor 14" monocr. basculante bifreq., 1 parallela, 2 seriali

TUTTO A LIRE 1.850.000

AT 286 20MHz

80286 26MHz operativi, 1Mb RAM esp. 8Mb su Piastra M/B NEAT-EMS gestore EMS Shadow RAM per Bios, controller 2FD 2HD, Floppy 1,2Mb o 1,44Mb HD 20 Seagate veloce 3,1/2, Monitor 14" monocr. basculante bifreq., 1 parallela, 2 seriali

TUTTO A LIRE 1.950.000

386sx

80386sx 16MHz 0WS, 1Mb RAM esp. 8Mb su piastra madre, quarzo aggiuntivo per 22MHz, controller per 2FD e 2HD, Floppy da 1,2Mb o 1,44Mb, HD 20Mb Seagate veloce 3,1/2, Tastiera 101 tasti Italiana, Scheda video Duale, Monitor 14" monocr. basculante bifreq., 2 seriali, 1 parallela.

TUTTO A LIRE 2.150.000

386 20MHz

80386 20MHz 0WS, 1Mb RAM esp. 4Mb su Piastra madre, controller per 2FD e 2HD, Floppy da 1,2Mb o 1,44Mb, HD da 20Mb Seagate veloce 3,1/2, Tastiera Italiana 101 tasti, Scheda video Duale, Monitor 14" monocr. basculante bifreq., 1 parallela, 2 seriali

TUTTO A LIRE 2.750.000

PERSONAL COMPUTER
Asem - Epson - Apple - Compaq
Sharp - Toshiba
Wyse

MiniComputers Honeywell

386 25MHz

80386 25MHz 0WS, 32Kb cache memory, 1Mb RAM esp. 8Mb su Piastra madre, controller per 2FD e 2HD, Floppy da 1,2Mb o 1,44Mb, HD da 20Mb Seagate veloce 3,1/2, Tastiera Italiana 101 tasti, Scheda video Duale, Monitor 14" monocr. basculante bifreq., 1 parallela, 2 seriali

TUTTO A LIRE 3.300.000

386 33MHz

80386 33MHz 0WS, 32Kb cache memory esp., 1Mb RAM esp. 8Mb su Piastra madre, controller per 2FD e 2HD Floppy da 1,2Mb o 1,44Mb, HD da 20Mb Seagate veloce 3,1/2, Tastiera Italiana 101 tasti, Scheda video Duale, Monitor 14" monocr. basculante bifreq., 1 parallela, 2 seriali

TUTTO A LIRE 3.950.000

486 25MHz

80486 25MHz 0WS Landmark 150MHz cache memory 4Mb RAM controller 2FD 2HD Floppy 1,2Mb o 1,44Mb HD 80 3,1/2 16ms, Tastiera italiana 101 tasti, Scheda video duale, Monitor 14" monocr. basculante bifreq.

TUTTO A LIRE 8.800.000

SU TUTTI I COMPUTERS:

HD 40Mb veloce + LIRE 250.000
FD 1.44Mb Epson + LIRE 150.000

VGA 800x600 8bit 256K + monitor 14" monocr. f.b. 16 toni di grigio
+ LIRE 250.000

VGA 1024 x 768 16 bit 512Kb + Monitor a colori 14" 1024 x 768 **+ LIRE 750.000**
HD 80 veloce + LIRE 500.000

STAMPANTI

STAR LC24-10	L. 599.000
STAR LC24-15	L. 899.000
NEC P2200	L. 599.000
NEC P7+	L. 1.500.000
EPSON (tutti i modelli)	telefonare
HONEYWELL	telefonare
PANASONIC (tutti i modelli)	telefonare
PANASONIC 11 p.p.m.	telefonare
STAR LASER 8	L. 2.999.000
XEROX 4030 11 p.p.m.	telefonare

PLOTTERS

ROLAND DXY-1100	L. 1.550.000
BENSON-OCE	telefonare
HOUSTON INST	telefonare

VARIE

VGA 800x600 8bit 256Kb	L. 190.000
VGA 1024x768 16bit 512Kb	L. 320.000
Tavola Graf. 12x12 Genius	L. 580.000
Mouse Logitech-Microsoft	telefonare
Scanman Logitech	telefonare
Modem GVC	da L. 150.000
VGA 1Mega	L. 700.000

TELEFAX
Canon FAX-80

Formato A4 omologato alimentatore da 5 documenti G3 compatibile 220V 13W polling, alta risoluzione, massima facilità d'uso.

OFFERTA LIRE 1.300.000

SOFTWARE

DOS - UNIX - XENIX - APPLE
*Borland *Ashton-tate
*Microsoft *Lotus
*Samna *Life Boat
*Digital Research *Ast

PROGETTO IDEA

Settore DTP

Finesse Logitech	L. 400.000
Page Maker	telefonare
Ventura	telefonare

GESTUDIO

Potente Software per la gestione delle pratiche dello Studio Legale: Citazioni, Ricorsi, Esecuzioni, Infortuni, Stragiudiziali, Parcellazione, Statistiche, Scadenzario delle udienze, Stesura atti giudiziari.

POSSIBILITA' DI LEASING

CONDIZIONI DI VENDITA
Tutti i prezzi si intendono IVA esclusa
Spedizione con corriere in tutta Italia
Garanzia 12 mesi totale

Rivenditore Autorizzato
PEGASO INFORMATICA
Via Mamurra, 35 - Formia (LT)
Tel. 0771/770751-267195

Agente per la zona di Viterbo
GENTILI PAOLO
Tel. 0761/458125

LOW PRICE LEADER

Ware Bit

Viale dell'Umanesimo, 80 - 00144 Roma
Tel. 06/ 592.19.77 - 592.19.78 - Fax 06/592.19.69