**Parte I - Capitolo 6 - La Memoria Di Massa E Le Altre Periferiche**

I dispositivi di I/O sono molto diversi tra loro, ma si possono clasisficare come segue: comportamento (input, output, memoria), partner (fornisce i dati da leggere o legge i dati inviati), velocità di trasferimento dei dati (la velocità con cui vengono trasmesse le informazioni tra il dispositivo e la memoria o il processore). Per gli utenti è importante che i dispositivi di memoria siano affidabili, attendibili e disponibili. Un dispositivo è ***affidabile*** se la qualità dei servizi forniti è tale da giustificare la fiducia in questi servizi, cioè se i servizi richiesti sono erogati in modo corretto e se il servizio viene interrotto quando differisce dalle specifiche definite (es. malfunzionamento). In questo caso il dispositivo dovrà essere ripristinato. L'***attendibilità*** è una misura della continuità con cui viene fornito il servizio a partire da un certo istante, cioè il tempo che intercorre prima che si verifichi un guasto. Una misura di attendibilità è il ***tempo medio di funzionamento prima del guasto*** (***MTTF***, ***Mean Time To Failure***); una misura collegata alla precedente è la ***frequenza annua media dei guasti*** (***AFR***, ***Annual Failure Rate***) che rappresenta la percentuale di dispositivi che riportano un guasto in un anno. L'interruzione di un servizio è misurata attraverso il ***tempo medio di riparazione*** (***MTTR***, ***Mean Time To Repair***), mentre il ***tempo medio tra due guasti*** (***MTBF***, ***Mean Time Between Failure***) è semplicemente la somma di MTTF e MTTR. La ***disponibilità*** è una misura relativa della corretta erogazione di un servizio ed è data dall'alternanza delle condizioni di erogazione corretta e interruzione del servizio, quindi definibile attraverso la formula:

Per migliorare il MTTF si può evitare il guasto, cioè costruire un dispositivo in modo da evitare il verificarsi del guasto, tollerare il guasto, cioè utilizzare la ridondanza per fare in modo che il servizio soddisfi le specifiche anche al verificarsi del guasto, o prevedere i malfunzionamenti, cioè prevedere la possibilità che si generino dei malfunzionamenti del dispositivo e sostituire il componente prima che sia guasto.

I ***dischi magnetici*** giacciono su un piatto rotante ricoperto da uno strato di materiale magnetico e utilizzano una testina mobile di scrittura e lettura per l'accesso ai dati. Un disco magnetico è formato da un gruppo di *piatti* che ruotano solidali a una velocità compresa tra i 5400 e i 15000 giri al minuto. Ciascun piatto memorizza su entrambe le facce e la sua superficie è suddivisa in cerchi concentrici chiamati *tracce* (*tracks*), che sono tipicamente da 10000 a 50000 per ogni lato del piatto. Ogni traccia è a sua volta suddivisa in *settori* (*sector*) i quali sono l'unità elementare di memorizzazione delle informazioni; ciascuna traccia contiene da 100 a 500 settori tipicamente di dimensione pari a 512 byte. La sequenza che viene registrata è costituita dal numero del settore seguito da uno spazio vuoto di separazione, a sua volta seguito dai dati memorizzati nel settore che comprendono il *codice di correzione degli errori* (*ECC*, *Error Correction Code*). Le testine di lettura e scrittura associate a ognuna delle due superfici di tutti i piatti sono collegate assieme e si muovono congiuntamente in modo che ogni testina si trova congiuntamente in corrispondenza della stessa traccia su tutte le superfici. L'insieme delle tracce dei diversi piatti che si trovano sotto le testine in un dato intervallo di tempo è detto *cilindro*. Per accedere ai dati il sistema operativo deve posizionare la testina sulla traccia giusta (***seek***). Il tempo necessario per compiere quest'operazione è detto ***tempo di ricerca*** (***seek time***). Il tempo di attesa che la testina deve attendere prima che il settore desiderato le passi sotto è detto ***latenza di rotazione*** o ***ritardo di rotazione***. Il tempo impiegato per trasferire un blocco di bit è detto ***tempo di trasferimento***. Il controllo dettagliato del disco e dei trasferimenti è gestito da un ***controllore del disco*** (***disk controller***) che aggiunge un'ultima componente di tempo di accesso totale al disco detta ***tempo di controllo*** che tiene conto del tempo aggiuntivo richiesto affinchè il controllore svolga il suo lavoro. Il tempo medio impiegato per realizzare un'operazione di I/O consiste nella somma di queste quattro componenti, più l'eventuale tempo di attesa di altri processi che stanno utilizzando il disco.

Le ***memorie flash*** rappresentano il primo vero concorrente dei dischi magnetici. Queste sono memorie a semiconduttore non volatili e hanno una latenza di circa 100-1000 volte inferiore ai dischi magnetici. Inoltre sono più piccole, hanno una maggiore efficienza energetica, maggiore resistenza agli urti, sono disponibili con capacità di memoria inferiori e per questo sono largamente utilizzate nei cellulari, nelle videocamere e nei lettori MP3. Una memoria flash è un tipo di memoria cancellabile elettricamente e programmabile a sola lettura (EEPROM, Electrically Erasable Programmable Read Only Memory). La prima memoria flash fu chiamata ***NOR Flash*** perchè era simile alle celle di memoria realizzate con porte NOR standard. Fu seguita pochi anni dopo dalla memoria flash ***NAND Flash*** che offriva una densità maggiore di memorizzazione delle informazioni, anche se la memoria poteva essere letta o scritta solamente in blocchi. Queste ultime memorie avevano un prezzo nettamente inferiore rispetto alle prime. Il limite delle memorie flash è dato dal fatto che queste si esauriscono dopo un certo numero di scritture. La maggior parte dei dispositivi che utilizzano memorie flash NAND contengono un controllore che distribuisce le scritture rimappando i blocchi di memoria che sono stati scritti più spesso sui blocchi che sono stati scritti di meno. Questa tecnica chiamata ***livellamento dell'usura*** (***wear leveling***) permette che il limite delle scritture per singola cella di memoria flash non venga esaurito. Tuttavia questa tecnica è valida per i dispositivi embedded e, quindi, non è stato risolto del tutto il problema dell'esaurimento della memoria, motivo per cui le memorie flash non sono popolari nei calcolatori desktop e server. Esiste una combinazione dei dispositivi detti ***dischi rigidi ibridi*** che possono contenere gigabyte di memoria flash, in modo tale che un portatile possa essere avviato più velocemente e risparmiare energia, consentendo al disco di rimanere inattivo più frequentemente.

In un'architettura i vari sottosistemi devono essere collegati attraverso interfacce di comunicazione. Per molti anni questi collegamenti sono stati realizzati tramite ***bus***, cioè dei canali di comunicazione condiviso che utilizza un unico insieme di fili per collegare più sottoinsiemi. I vantaggi di un'organizzazione a bus sono la versatilità e il basso costo, mentre gli svantaggi sono la creazione di un collo di bottiglia nella comunicazione tra i dispositivi con il rischio di limitare la velocità massima di trasferimento dell'I/O. I bus vengono tradizionalmente suddivisi in due categorie: i ***bus processore-memoria*** sono corti, di solito ad alta velocità e devono adattarsi alle caratteristiche del sistema di memoria in modo da massimizzare la larghezza di banda tra processore e memoria; i ***bus di I/O*** possono essere lunghi, permettono di collegare molti tipi di dispositivi e spesso la larghezza della banda dei dispositivi può assumere un ampio spettro di valori. Questi ultimi non si interfacciano direttamente con la memoria, ma utilizzano un bus processore-memoria o un generico bus di sistema (backplane bus) per collegarsi alla memoria. Una delle ragioni per cui la progettazione di un bus è molto difficile è che la sua velocità massima è limitata in gran parte da fattori fisici (lunghezza del bus e numero di dispositivi collegati). L'industria è passata dunque all'utilizzo di bus paralleli condivisi a collegamenti seriali punto-a-punto, ad alta velocità e con interruttori. Una *transazione* è costituita dall'invio dell'indirizzo e dalla spedizione o ricezione dei dati. Una transazione può essere di lettura se i dati passano dalla memoria al processore o, viceversa, di scrittura. I bus tradizionali sono sincroni, cioè hanno una linea per la trasmissione del segnale di clock e un protocollo di comunicazione prefissato agganciato al clock. La trasmissione dei dati tra il mittente e il destinatario è coordinata da un protocollo detto ***protocollo di handshaking***, che consiste in una serie di passi tali che il mittente e il destinatario procedano al passo successivo solo quando entrambe le parti sono d'accordo. I bus sincroni hanno lo svantaggio che i dispositivi collegati devono funzionare tutti alla stessa frequenza di clock e, inoltre, non possono essere lunghi se devono essere veloci. Questi problemi hanno portato allo sviluppo dei bus asincroni che non richiedono il segnale di clock. Nei processori x86, il processore è connesso alle periferiche attraverso due chip principali: il *north bridge* è più vicino al processore e ha il compito di connettere e controllare la memoria; il *south bridge* è connesso al north bridge e ha la funzione di connettere i dispositivi di I/O e di controllarli. Il north bridge è sostanzialmente un controllore DMA che connette il processore alla memoria, a una scheda grafica e al south bridge. Il south bridge connette il north bridge ai bus di I/O.

Vediamo ora come una richiesta di I/O viene trasformata in un comando. Il sistema operativo gioca un ruolo cruciale in questo meccanismo in quanto esso costituisce l'interfaccia tra l'hardware e il programma che richiede l'operazione di I/O. Per inviare un comando a un dispositivo di I/O il processore deve poter indirizzare un dispositivo e fornirgli il comando su una o più parole. Questo può avvenire attraverso due metodi: ***I/O mappato in memoria*** (***memory-mapped I/O***), ***istruzioni di I/O speciali***. Le istruzioni speciali vengono utilizzate per inviare un comando a un dispositivo di I/O specificando sia il numero del dispositivo sia la parola contenente il comando o, in alternativa, la locazione di memoria che contiene il comando da eseguire. Nel caso di memory-mapped, alcune aree dello spazio di indirizzamento vengono assegnate ai dispositivi di I/O, quindi, una lettura o una scrittura in questi indirizzi di memoria corrisponde a un comando rivolto a un certo dispositivo I/O. Ai programmi utente non è consentito eseguire direttamente le operazioni in quanto questi indirizzi sono protetti da un procedimento di traduzione degli indirizzi gestito dal sistema operativo. Può accadere inoltre che il processore debba verificare lo stato del dispositivo per controllare le attività associate che intercorrono tra due intervalli di tempo. Il registro *Stato* contiene un bit di fine operazione che viene impostato a uno se il dispositivo ha terminato la sua funzione e un bit di errore che verifica che non ci siano stati problemi durante il procedimento richiesto al dispositivo. Il controllo periodico del bit di stato per verificare se è possibile passare all'operazione di I/O successiva è detto ***polling***. Il polling è il metodo più semplice per far comunicare un dispositivo di I/O con il processore: il dispositivo scrive informazioni nel registro di stato, mentre il processore le legge e, dopo averle controllate, preleva la informazioni. Lo svantaggio del polling è che può far perdere molto tempo al processore che è molto più veloce del dispositivo di I/O (es. il processore legge il registro di stato tante volte prima che il dispositivo cambi il suo stato al termine di un'operazione molto lenta). Per risolvere il problema legato al tempo di esecuzione del polling, si è passato all'***I/O controllato da interrupt*** (***interrupt-driven I/O***); questa tecnica impiega gli interrupt per segnalare al processore che un dispositivo di I/O ha bisogno della sua attenzione in quanto ha terminato qualche operazione. Un interrupt di I/O si comporta come un'eccezione: è asincrono rispetto all'esecuzione delle istruzioni, trasmette, oltre a una notifica di verifica dell'interrupt, anche altre informazioni come l'identità del dispositivo che l'ha generato. Il meccanismo di interrupt elimina la necessità di interrogazione periodica del processore, permettendo a quest'ultimo di concentrarsi sull'esecuzione dei programmi. Per gestire i vari dispositivi, questo metodo possiede diversi livelli di priorità che indicano l'ordine in cui il processore deve gestire gli interrupt. Tipicamente gli interrupt di I/O hanno priorità bassa rispetto alle eccezione generate internamente al processore. Il registro di stato contiene un bit di abilitazione degli interrupt che consente ai dispositivi il verificarsi o meno degli interrupt. Un modo più raffinato per bloccare gli interrupt è rappresentato dalla *maschera degli interrupt* che consiste in un gruppo di bit, ciascuno dei quali corrisponde a un bit del campo degli interrupt precedenti del registro causa. Per abilitare un certo interrupt deve essere presente un bit 1 in corrispondenza della maschera degli interrupt. I due metodi visti per la comunicazione tra il processore e il dispositivo di I/O consentono il trasferimento dei dati tra il dispositivo e la memoria in due modi diversi. Il metodo basato sul polling consente al processore di caricare i dati dai registri del dispositivo di I/O e salvarli in memoria. Per quanto riguarda il metodo basato sugli interrupt, il sistema operativo trasferisce ancora piccole quantità di dati da e verso il dispositivo di I/O. Quando si verifica un interrupt il sistema operativo legge lo stato del dispositivo per verificare che non ci siano stati errori. Per i dispositivi a banda larga (dischi rigidi) il trasferimento riguarda blocchi di dati relativamente grandi, quindi i progettisti hanno inventato un meccanismo per scaricare il processore e far si che il controllore del dispositivo trasferisca i dati direttamente in memoria. Questo meccanismo è detto ***accesso diretto alla memoria*** (***DMA***, ***Direct Memory Access***). Il meccanismo di interrupt viene ancora utilizzato, ma solo per segnalare al processore che il trasferimento ha avuto esito positivo o negativo. Il DMA è implementato mediante un controllore specializzato detto *master della connessione* che trasferisce i dati tra dispositivo e memoria, indipendentemente dal processore. Il trasferimento di DMA avviene attraverso tre fasi: il processore inizializza il DMA fornendo l'identità del dispositivo, l'operazione che deve eseguire, l'indirizzo di memoria, i dati da trasferire e il numero di byte; il DMA inizia l'operazione sul dispositivo e governa la connessione (il DMA fornisce l'indirizzo di memoria per singola lettura o scrittura; se il controllo va a buon fine si passa all'elemento successivo); terminato il trasferimento, il controllore interrompe il processore che verifica se l'intera operazione è stata completata con successo. Con l'utilizzo del DMA si crea un percorso alternativo che non passa attraverso il meccanismo di traduzione degli indirizzi o la gerarchia della cache. Questa differenza genera alcuni problemi nei sistemi che utilizzano la memoria virtuale o in quelli dotati di cache. Le difficoltà nell'avere un DMA in un sistema dotato di memoria virtuale nasce dal fatto che le pagine hanno un indirizzo sia fisico che virtuale. Nei sistemi dotati di cache, invece, i problemi risultano dal fatto che possono esistere più copie dello stesso dato (nella cache e nella memoria). Dato che il meccanismo di DMA invia dati direttamente alla memoria, il contenuto di una locazione di memoria vista dal processore può essere diverso dal contenuto visto dal DMA. Se il DMA scrive i dati in locazioni di memoria il cui contenuto si trova anche nella cache, il processore leggerà il vecchio valore. I problemi legati alla lettura di valori più recenti dei dati non ancora scritti in memoria è detto ***problema dei dati stantii*** o ***problema della coerenza***.

Per progettare un sistema di I/O bisogna considerare i vincoli sulla latenza e i vincoli sulla larghezza di banda. In entrambi i casi, la conoscenza del tipo di traffico influenza la progettazione e l'analisi. I vincoli sulla latenza implicano che la latenza richiesta per completare un'operazione di I/O sia inferiore a un certo intervallo di tempo. L'approccio generale alla progettazione consiste nelle seguenti fasi: identificare il collegamento più debole del dispositivo I/O, configurare il componente per sostenere la larghezza di banda richiesta, determinare i requisiti del resto del sistema.

Il parallelismo deve coinvolgere anche i sistemi di I/O oltre alla parte di calcolo, altrmenti lo sforzo speso nel rendere parallela l'esecuzione verrebbe vanificata ogni volta che un programma esegue un'operazione di I/O. L'aumento della velocità dell'I/O fu la motivazione principale che portò all'introduzione degli ***insiemi di dischi*** (***disk array***). Dagli anni ottanta fu proposto di sostituire i dischi grandi con dischi più piccoli per aumentare le prestazioni, grazie alla presenza di un numero maggiore di testine lettura/scrittura. Questo cambiamento ha avuto successo anche per i processori multipli, poichè consente di supportare molti accessi indipendenti e di trasferire grandi quantità di dati accedendo a dischi diversi. Gli insiemi di dischi sono più vantaggiosi anche in termini di costo, di consumo energetico e di dimensioni. Lo svantaggio è rappresentato dal fatto che la presenza di dischi rischiava di rendere l'affidabilità del dispositivo molto più bassa (MTTF molto basso). La soluzione è stata trovata aggiungendo la ***ridondanza*** in modo da far gestire al sistema i guasti del disco senza perdere informazioni. Una maggiore affidabilità è più facilmente ottenibile tramite un insieme di dischi ridondanti poco costosi detti ***reduntant array of inexpensive disks*** (***RAID***, ***insieme ridondante di dischi economici***). Esistono diversi gradi di ridondanza necessaria. È possibile ottenere un RAID 0, cioè nessuna ridondanza, semplicemente suddividendo i dati su più dischi, in modo da forzare l'accesso in modo automatico. L'operazione denominata ***stiping*** su un insieme di dischi, fa sì che il disco RAID appaia al software come un unico grande disco. Utilizzando il doppio dei dischi di un RAID 0 per compensare i guasti si ha uno schema chiamato ***mirroring*** o ***shadowing***; ogni volta che un dato viene scritto su un disco, questo viene scritto anche sul disco ridondante. Se il disco subisce un guasto il sistema accede al disco mirror per recuperare l'informazione. Questo tipo di RAID 1 è il più costoso. Il RAID 2 adotta un codice di riconoscimento dell'errore molto utilizzato nelle memorie, ma, ormai, caduto in disuso. Nel RAID 3 non viene creata una copia completa dei dati originali, ma vengono aggiunte in un disco le informazioni ridondanti sufficienti per recuperare i dati persi a causa di un eventuale guasto. Il RAID 4 utilizza lo stesso principio del RAID 3, ma la modalità di accesso ai dati è diversa in quanto la memorizzazione è associata a un insieme di blocchi (nel RAID 3 ogni accesso avviene su tutti i dischi). Un inconveniente di questo schema deriva dalla necessità di aggiornare il disco di parità a ogni scrittura, in modo da renderlo un collo di bottiglia per le scritture in sequenza. Nel RAID 5 il problema viene eliminato distribuendo l'informazione su tutti i dischi in modo da evitare che esista un solo collo di bottiglia per le scritture. I RAID visti sono molto utilizzati nei server; il loro punto debole è la riparazione. Per evitare di rendere i dati inaccessibili bisogna che l'insieme dei dischi sia realizzato in modo tale da permettere la sostituzione dei dischi danneggiati senza dover spegnere l'intero sistema. I RAID hanno ridondanza sufficiente per continuare a funzionare, ma la ***sostituzione a caldo*** (***hot swapping***) pone requisiti fisici ed elettrici particolarmente restrittivi. Alcuni dispositivi contengono una ***copia di scorta*** che consiste in una componente hardware di riserva che può prendere immediatamente il posto di una componente che si è guastata.