13 Giugno 2023

ARCHITETTURE DI CALCOLO LEZIONE 25

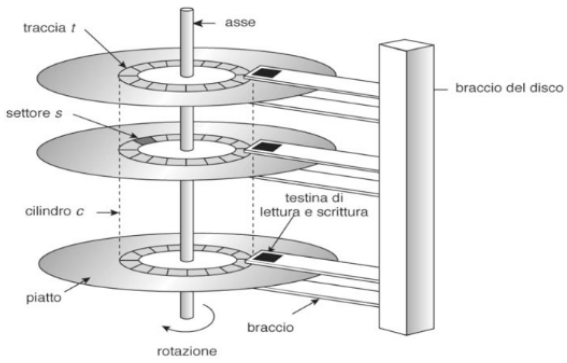
Gestione della memoria secondaria, gestione del disco e RAID

GESTIONE DELLA MEMORIA SECONDARIA

In questa lezione verrà affrontato il tema di come il sistema operativo gestisca concretamente l'accesso al disco. La gestione del disco è tra le più eterogenee dei sistemi operativi; infatti, è caratterizzata da una varietà di soluzioni.
Il sistema operativo deve essere in grado da un lato di gestire l'eterogeneità, dall'altro fornire alle componenti che stanno di sopra una vista uniforme.
Deve, inoltre, ottimizzare l'accesso ai dati. Ad esempio, quando si è parlato di un file, si è detto essere composto da una serie di blocchi che devono essere letti o scritti sul disco.
Se avessimo un file costituito da tre blocchi, questi potrebbero trovarsi in qualsiasi punto del disco quindi anche lontani tra loro. Per leggerli, la testina del disco dovrà andare nel punto in cui il dato si trova per leggerlo. È un po' come il corriere fa con dei pacchi. Potrebbe prendere i pacchi uno per volta, leggere l'indirizzo e andare a consegnare. Facendo in questo modo, però, non si ottimizza il percorso.
Al fine di ottimizzare l'I/O il sistema operativo dispone di un ventaglio di algoritmi da usare in contesti differenti.

RICHIAMO DISCHI MAGNETICI

I dischi magnetici sono costituiti da piatti, anticamente in alluminio e che ora sono in vetro ricoperto da uno strato magnetizzato; sono
piatti più sottili con superficie più regolare. La testina poi attraverso l'induzione legge o scrive su materiale magnetico.
Si tratta di una struttura con una serie di piatti che girano attorno ad un asse di rotazione; a distanza costante dall'asse di rotazione sono presenti dei blocchi che formano una traccia. Più tracce alla stessa distanza dall'asse sullo stesso piano, formano un cilindro.



Per poter leggere in qualsiasi punto del disco
sono necessari due movimenti:
•Seek time: è un movimento atto a spostare le testine sul cilindro del settore interessato;
•Rotational latency: tempo di rotazione del disco per portare le testine sul settore.

Tra questi due tempi quello che costa di più è il seek time, in quanto il posizionamento della testina è un movimento che deve essere fatto 'ad hoc', mentre la rotazione è qualcosa che si verifica sempre.
Dal punto di vista logico il disco possiamo vederlo come un array di blocchi, numerati da 0 fino ad n. Serve una convenzione che ci dica qual è il settore 0 (primo settore della prima traccia del cilindro più esterno) e quale il blocco n (quello finale, ossia quello del cilindro più interno).

PERCHÉ SERVE FARE LO SCEHDULING DEL DISCO?
Questa procedura nasce con l'obiettivo di ridurre il tempo di accesso ai dati e per aumentare l'ampiezza di banda (numero di dati trasferiti per unità di tempo). Si può agire su due parametri: seek time e rotazional latency (tempo di rotazione).

SEMPLIFICAZIONE: definiremo che il seek time è approssimativamente equivalente alla distanza di seek, ossia la distanza percorsa dalla testina per raggiungere il settore di interesse. Dovremmo misurare il seek time con un cronometro; invece diremo che il seek time è il tempo che impiega la testina per muoversi; questo è proporzionale allo spazio percorso. L'algoritmo di scheduling minimizza dunque la distanza di seek, e non il tempo di seek. Sono simili ma non uguali.
La differenza tra un settore e l'altro è irrilevante per questi algoritmi. Si fa riferimento solo al numero di cilindri.

Per valutarli usiamo la seguente sequenza di cilindri su cui si trovano i settori richiesti:

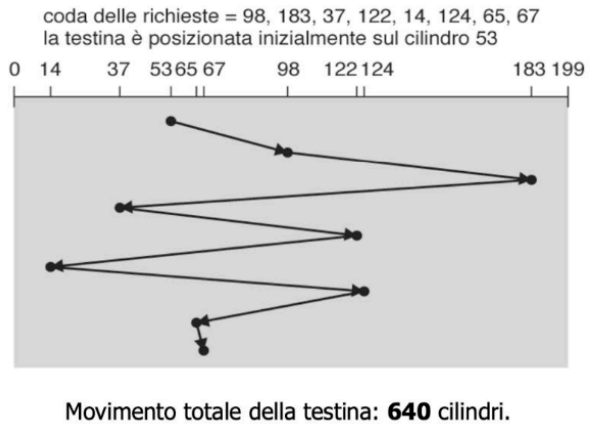
98, 183, 37, 122, 14, 124, 65, 67

Posizione iniziale della testina: ***53***

Supponiamo di dover accedere a dei blocchi posizionati sui cilindri in figura. Dobbiamo sapere dove si trova inizialmente la testina. In questo caso al punto 53. È fondamentale conoscere la posizione della testina. Per alcuni algoritmi è inoltre importante anche conoscere la direzione della testina.

SCHEDULING FCFS

Utilizza la tecnica FIFO.

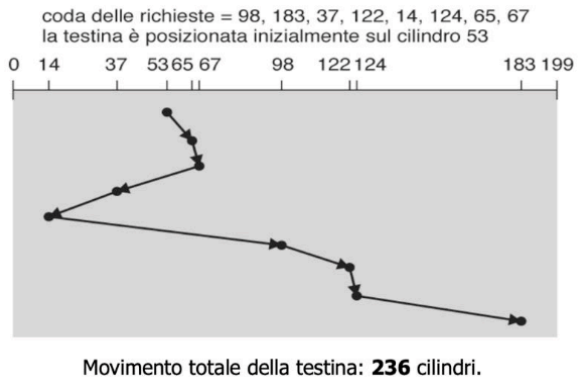


Movimento totale della testina: **640** cilindri.

Dalla posizione 53, la testina serve le richieste in ordine di arrivo. Si va da 53, a 98, a 183, a 37, a 122, 14, 124, 65, 67. Questa pianificazione è inefficiente perché la testina va di qua e di là senza un ordine preciso.
Per misurare lo spazio percorso si fa la somma dei segmenti. Quindi ad esempio, 183 – 53, 183 – 37, 122 - 37, 122 – 14, 124 – 14, 124 – 65, 67 – 65 = 130 + 146 + 85 + 108 + 110 + 59 + 2 = 640 cilindri.

SCHEDULING SSTF (Short seek time fisrt)

Fra le richieste pendenti si va a servire quella più vicina alla posizione corrente. È una forma di scheduling SJF. Si sceglie il cilindro con il più piccolo percorso.

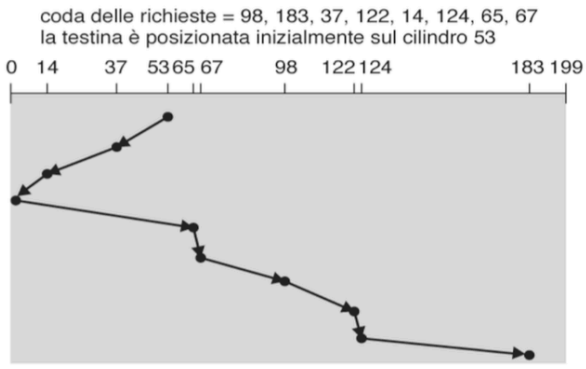


Movimento totale della testina: **236** cilindri.

Può causare starvation, ossia se arriva un processo più breve rispetto a quello in attesa, passa prima quello e gli altri aspettano.
In questo caso dalla posizione 53, si va alla 65, poi 67, poi 37, 14, 98, 122, 124, 183.
Anche in questo caso per calcolare lo spazio percorso si fa la somma dei segmenti: 67 – 53, 67 – 14, 124 – 14, 183 – 124 = 14 + 53 + 110 + 59 = 236 cilindri Riduce di quasi 1/3 il tempo totale rispetto la FCFS.

SCHEDULING SCAN (ALGORTIMO DELL'ASCENSORE)

Un ulteriore algoritmo, tra i più usati nella realtà, è lo SCHEDULING SCAN (algoritmo dell'ascensore).

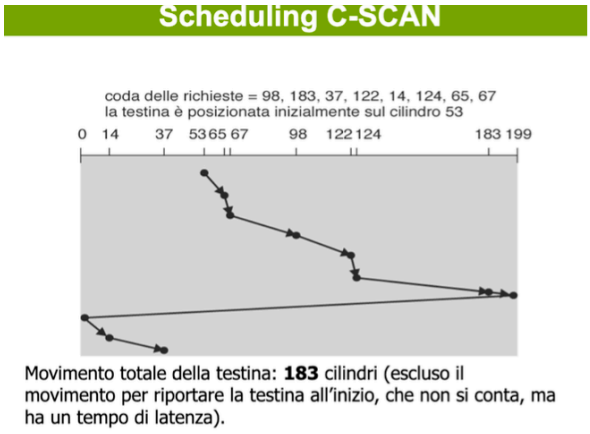


Movimento totale della testina: **236** cilindri.

In questo caso si fa partire la testina da una parte del disco, e questa va fino all'estremo opposto servendo tutte le richieste dei blocchi che si trovano lungo il percorso. Poi viene invertita la marcia facendo lo stesso nell'altra direzione.
Inizialmente la testina va dal 53 allo 0 (verso sx); poi va nella direzione opposta verso dx.
Il movimento totale della testina resta comunque di 236 cilindri.
Un limite di questo algoritmo è il fatto che nella media lavora bene ma ha una varianza piuttosto accentuata: ci sono valori che vengono serviti in tempi migliori rispetto ad altri che devono attendere molto di più rispetto agli altri algoritmi.
Per porre rimedio a tale problema si propone la tecnica C-SCAN (Circular-SCAN)

SCHEDULING C-SCAN

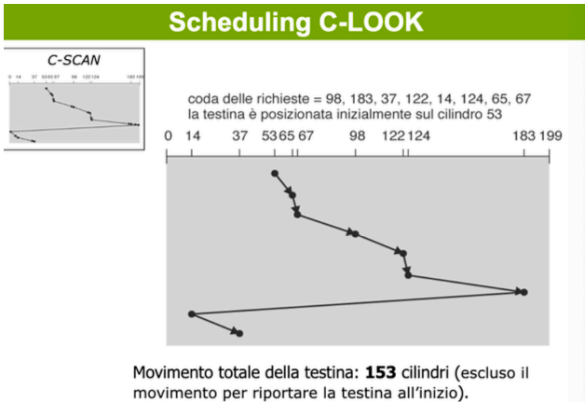
Variante dell'algoritmo SCAN che offre un tempo di attesa più uniforme.
La testina parte da un estremo del disco e muovendosi fino all'altro estremo serve tutte le richieste di blocchi che si trovano lungo il percorso, quindi ritorna all'altro estremo del disco (senza servire le richieste nello spostamento) per ripartire da lì.



L'azzeramento della testina è un'operazione estremamente rapida tant'è che il numero di cilindri attraversati non viene considerato nella misurazione della distanza di seek.
In tale algoritmo, quindi, la coda di richiesta viene considerata come una lista circolare in cui all'ultimo elemento della lista segue il primo elemento in coda.
Inizialmente la testina va da 53 a 199 (verso destra), svolgendo tutte le operazioni richieste in questa direzione. Arrivata qui, la testina compie uno "scatto" e viene riposizionata velocemente nella posizione 0 (verso sinistra).
NOTA: il moto della testina prevede tre fasi – accelerazione, crociera e decelerazione; anche durante questo "scatto" le fasi rimangono sempre queste.
Dalla posizione 0, la testina si muove nuovamente verso destra fino ad esaurire le richieste. Il movimento totale della testina si è ridotto: 183.

SCHEDULING LOOK E C-LOOK

Lo scheduling LOOK e C-LOOK sono variante rispettivamente di SCAN e C-SCAN. La testina viene spostata non fino alla fine del disco ma solo fino a che ci sono richieste in quella direzione (quindi l'ultimo cilindro da servire in quella direzione). LOOK indica il fatto che si "guarda" le posizioni terminali a cui la testina deve arrivare.



Nell'esempio, se so che l'ultima richiesta in termini di posizioni è in posizione 183, anziché far scorrere la testina fino alla fine del disco (199) la faccio arrivare a 183 per poi farla tornare immediatamente verso l'algoritmo C-SCAN, nello spostamento verso destra vengono esaurite le richieste che vengono sinistra, fino alla posizione della richiesta "minima" (più a sinistra) invece che a zero, in questo caso in posizione 14. Come per presentate alla testina.

Il professore non presenta un esempio per l'algoritmo LOOK ma è analogo all'algoritmo SCAN con la differenza che la testina scorre fino alla posizione della richiesta massima e poi torna indietro fino alla posizione della richiesta minima.
Il movimento totale si è ulteriormente ridotto a 153 cilindri.
Tra tutti questi algoritmi, quelli più diffusi nei S.O. general purpose sono SCAN e C-SCAN.
L'SSTF con tecniche di prevenzione della starvation (aging, e..) è usato comunemente, ma è sempre preferibile la tecnica più semplice per evitare eventuali errori.

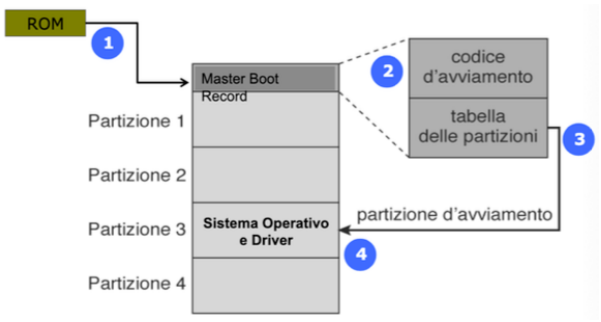
GESTIONE DEL DISCO

Accanto allo scheduling del disco, il S.O. ha altre funzioni di gestione del disco:

- Formattazione fisica e partizionamento: predisposizione del disco alla sua scrittura tramite la divisione in cilindri , tracce e settori. Questi marker (bit) servono alla testina per capire dove posizionarsi sul disco.
- Formattazione logica: creazione di un file system sul disco. In base al tipo di file system, i file saranno organizzati e visualizzati in modo diverso (con indicizzazione, FAT, ecc.).

Accanto al concetto di formattazione fisica (o di basso livello) e formattazione logica, troviamo il concetto di programma d'avvio situato nella ROM. Tale programma di boot ha il compito inizializzare il sistema tramite ricerca del S.O. stesso nel Master Boot Record nella ROM. Questo Master Boot Record "conosce" la partizione del disco (minimo una partizione, in genere come standard un disco ha 1 o 2 partizioni e la seconda partizione viene usata per la funzione di ripristino). Le partizioni possono anche essere create dall'utente tramite apposite applicazioni.

L'uso delle partizioni permette di installare su ciascuna di esse un file system diverso.



Ad esempio, su Linux una cosa che si fa spesso è creare due partizioni:

•Una generale in cui viene caricato il S.O. e tutti i programmi;

•Una detta di swap, in cui vengono caricati i file temporanei per la gestione della memoria virtuale.

In Windows questa distinzione tra partizione generale e di swapping non avviene e troviamo un'unica partizione contenente il tutto.

Ipotizzando di avere più partizioni, nel Master Boot Record sono contenuti il codice di avviamento e la tabella delle partizioni, che ci indica in quale partizione si trova il Sistema Operativo. Quindi, è un percorso a fasi:

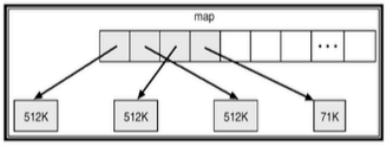
1. Caricamento del programma di bootstrap nella ROM;
2. Il programma va nel Master Boot Record, dove viene fatto partire un piccolo codice di avviamento di un piccolo programma che possiamo codificare;
3. Il codice di avvio legge la tabella delle partizioni, individua la posizione nella memoria secondaria del S.O. e lo avvia.

GESTIONE DELLO SPAZIO DI SWAP

In tutti i S.O. moderni il disco viene utilizzato anche come spazio di swap, ossia come estensione della memoria centrale in cui avviene il caricamento/scaricamento di programmi e dati. Ci sono due soluzioni:

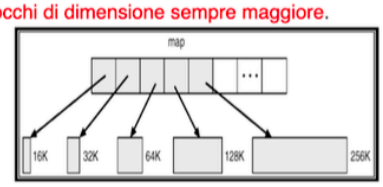
- Tenere i file di swapping come dei normali file di sistema all'interno del file system generale (WINDOWS).
- Separare lo spazio di swapping da quello generale (LINUX, UNIX BSD).

Lo **spazio di swap del segmento di testo** è allocato in blocchi di 512 K tranne per l'ultimo blocco.



Lo **spazio di swap del segmento dati** è allocato in blocchi di dimensione variabile (multipli di 16 K).

Per processi piccoli si usano blocchi piccoli, per processi grandi si allocano blocchi di dimensione sempre maggiore.



In UNIX 4.3BSD lo spazio di swap viene diviso in spazio di swap per i programmi (text segment) e per i dati (data segment). I codici e i dati hanno dimensioni tipiche per motivi legati all'efficienza dell'accesso dei dati. Nell'immagine a sinistra, si osserva la mappa di swap in UNIX 4.3BSD in cui al codice è riservato uno spazio di 512k (text segment) e ai dati di 16k (data segment). Mentre lo spazio di swap del segmento di testo è diviso in blocchi singoli (da 512k), nello spazio di swap del segmento dati si possono allocare dati in blocchi di dimensione multipla del blocco minimo (nell'esempio, 16k, 32k, 64k, ecc.).
NOTA: stiamo trattando di blocchi della memoria su disco, in cui un singolo blocco potrebbe contenere un enorme numero di frame (ad esempio 16k = 4 frame, 32k = 8 frame, ecc.).

DISCHI RAID

Quando serve memorizzare grandi moli di dati e/o mantenere una elevata affidabilità della memoria secondaria si può usare una batteria di dischi detta struttura RAID.
RAID: Redundant Array of Indipendent Disks o batterie ridondanti di dischi
Nei server i dischi basati su testina non vengono utilizzati da soli ma vengono messi in un plaid (contenitore), che ne contiene vari che vanno da due salendo esponenzialmente (2,4,8,16...) e funziona come una singola unità di memoria secondaria.
Si mettono in una configurazione ridondante, cioè un disco viene duplicato su un altro disco e si avranno due dischi uguali che in caso di rottura di uno dei due non si avrà perdita di dati.
C'è anche un altro vantaggio che è l'aumento delle prestazioni perché, se si ha un file presente su entrambi i dischi, si può leggere la prima parte del file da un disco e la seconda dall'altro disco diminuendo i tempi tenendo conto ovviamente anche della sincronizzazione delle testine.
Presentando questo "unico disco" all'utente vengono presentate anche delle funzionalità aggiuntive che sono dovute a 3 tecniche:

- Mirroring (un disco può essere la copia di un altro)
- Striping (un file viene memorizzato in parte su un disco e in parte su un altro)
- Bit di parità (ci permettono di capire su un insieme di bit se il file si è modificato)

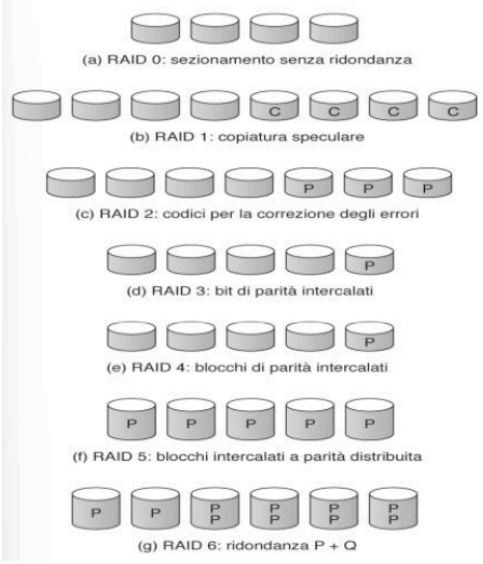
Mirroring
Con il mirroring si ottiene un grande aumento dell'affidabilità, in quanto se si hanno 2 dischi uguali qual è la probabilità che i due dischi si rompano nello stesso momento?
Sicuramente molto bassa e trascurabile però bisogna pensare anche ad eventi improvvisi come improvvisi cali di tensione e fulmini, danneggerebbero con tutta probabilità entrambi i dischi.

La frequenza con la quale si possono gestire le richieste di lettura raddoppia, poiché ciascuna richiesta si può inviare indifferentemente a uno dei due dischi (sempre che entrambi i dischi siano funzionanti, condizione che è quasi sempre soddisfatta). La capacità di trasferimento di ciascuna lettura è la stessa di quella di un sistema a singolo disco, ma il numero di letture per unità di tempo raddoppia.
L'osservazione della realtà, comunque, ci dice che il mirroring rimane la soluzione di prevenzione migliore in tutti i server.

Striping
Se abbiamo un disco raid composto da n dischi, il file viene diviso in n parti e ciascuna di queste n parti viene memorizzata in uno dei dischi.
Ovviamente questa tecnica rischia di far perdere il nostro file in quanto se uno dei dischi si rompe, il nostro file è perso.
Qual' è il vantaggio dello striping?
Poter leggere il file a velocità doppia, quadrupla e cosi via in quanto il file viene letto su dischi diversi.

Bit di parità
Se io ho n dischi, possono prendere i primi n-1 dischi per memorizzare i dati e poi utilizzare l'ultimo disco per utilizzare i bit di parità.
Quindi se un bit viene alterato sui dischi, non ci sarà più concordanza tra i dischi dei dati e i bit di parità quindi il sistema potrà dischi che i file sono stati corrotti.
Se si ha un singolo bit di parità possiamo solo rilevare gli errori mentre se avete di più, ad esempio 3, si può correggere anche l'errore.
Esistono delle tecniche di correzione che memorizzano più bit supplementari e possono ricostruire i dati nel caso di un singolo bit danneggiato.
Ovviamente queste tre tecniche, per diminuire gli svantaggi che hanno e prenderne tutti i vantaggi, vengono combinate.

Livelli Raid



I livelli raid partono da livello 0 e intende la tipologia di combinazione delle tecniche che viene adottata.
P indica i bit di correzione degli errori C indica una seconda copia dei dati)
RAID 0 applica uno striping a livello di blocchi, (divide i dati equamente tra due o più dischi) senza nessuna informazione di parità o ridondanza. Questo raid è finalizzato solamente a migliorare le prestazioni a discapito del fallimento per via della perdita dei dati.
RAID 1 crea una copia esatta (o mirror) di tutti i dati su più dischi.
Capacità aumentata di 4 e ridondanza aumentata di 2.

RAID 2 divide i dati al livello di bit invece che di blocco con dischi di parità.
RAID 3 usa lo striping livello di byte con un disco dedicato alla parità. RAID 4 usa lo striping a livello di blocchi con un disco dedicato alla parità.
RAID 5 usa lo striping a livello di blocco con i dati di parità distribuiti tra tutti i dischi appartenenti al RAID.
RAID 6 usa lo striping a livello di blocchi con i dati di parità distribuiti due volte tra tutti i dischi.
Esistono anche versioni intermedie, ad esempio raid 0+1 e 1+0, che sono semplicemente ricombinazini delle varie tecniche.
Del raid 3 4 5 6 il prof non ci chiederà la spiegazione ma potrebbe chiederci semplicemente quanti sono i raid standard (SONO 7). Mentre per i raid 0 1 2 ci chiede anche come sono disposti

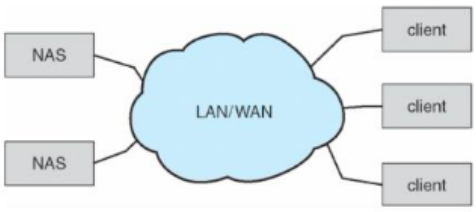
RAID 0 e 1

Il raid 0 è una tecnica di facile implementazione che garantisce prestazioni elevate, però non è affidabile.
Il raid 1 può esser visto come la controparte del raid 0, è orientato soprattutto alla tolleranza ai guasti, ma non è ottimale nella gestione dello spazio e ne migliora le prestazioni rispetto un singolo disco.

COLLEGAMENTO DEI DISCHI

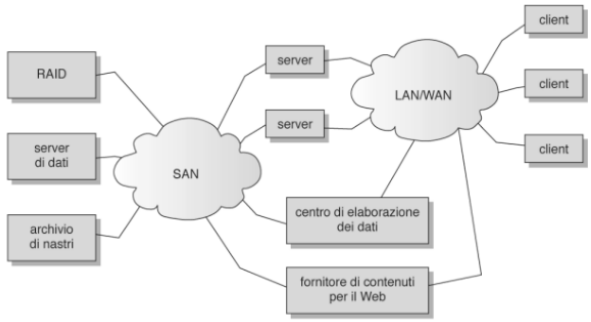
I dischi possono essere connessi in due modi principali:
- Collegati ad un host (PC/workstation) attraverso una porta di I/O.
- Collegati alla rete attraverso una connessione di rete

Uno storage disponibile su rete è Network-attached storage (Nas), viene utilizzato per denominare tutti i metodi che si utilizzano per collegare uno storage a una rete.
Nfs è un tipico protocollo Nas



Implementato mediante chiamate di procedura remota (RPCs) tra l'host e lo storage
Tempo fa si utilizzava sui modem e appariva sul nostro pc come "disco C" che intendeva un disco collegato al modem su cui si salvavano i propri dati.

Un altro concetto per gli storage di rete è il SAN (Storage-Area Network).



Sono reti specializzate nel trasferire dati ada alta velocità tra i client e i vari dispositivi
È comune in ambienti di storage di grandi dimensioni e più host sono collegati a più dispositivi di storage.

IMPLEMENTAZIONE DELLA MEMORIA STABILE

Se si vogliono implementare i dati il raid è una soluzione “online”, mentre per esser sicuri di avere un’implementazione stabile serve avere un backup offline.
La memoria stabile garantisce che non ci siano mai perdite di dati e per implementarla:

- Replicare l’informazione su più supporti di memoria non volatile che abbiano modalità di guasto differenti e indipendenti
- Aggiornare l’informazione in maniera controllata per assicurare che si possano ottenere i dati stabili dopo un fallimento:
    - sia durante il trasferimento dei dati
    - sia durante un ripristino

Soprattutto nell’ambito medico è importante preservare i dati, per evitare di subire attacchi e perdere dati importanti.
Quindi si devono sempre avere delle copie in delle memorie stabili, un esempio possono essere le memorie cloud che hanno vari vantaggi mentre ci sono dei contro come il non rispetto delle leggi sulla privacy quindi i dati medici non possono essere messi su un cloud.
Altro contro è che se i dati vengono modificati o cryptati, il cloud li salverà in quest ultima versione e quindi avremmo subito comunque un attacco informatico.
La memoria stabile per eccellenza rimane la memoria “offline” e che quindi non è mai collegata ad internet e non può subire attacchi.
Per evitare che i dati si corrompano bisogna anche ravvivare i dati ed è questo ciò che i cloud fanno in automatico per non per perdere i dati.