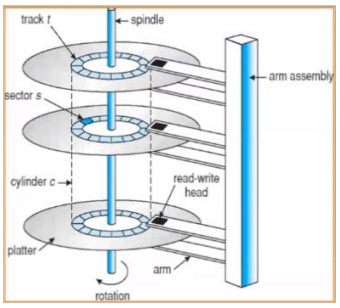
4.0 DISCO MAGNETICO

Per memoria di massa intendiamo la memoria secondaria la quale può essere associata ai vecchi nastri magnetici che possono contenere grosse quantità di dati, poco usato al giorno d'oggi. Il problema del nastro il tempo di accesso è sequenziale, di conseguenza si perde molto tempo.



Un disco magnetico è fatto da una serie di piatti impilati uno nell'altro che ruotano attorno ad un perno centrale e ognuno dei piatti è suddiviso in tracce che sarebbero i cerchi concentrici in didascalia ed ogni traccia è divisa a sua volta in settori. La lettura da un settore avviene attraverso una testina di lettura che si muove lungo il raggio dei piatti.

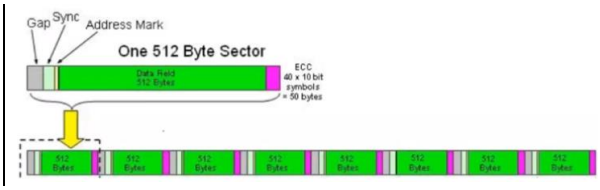
Quando per esempio si deve prelevare un blocco la prima cosa che bisogna fare è andare a muovere il braccio lungo il raggio per andare a mettersi nella posizione relativa alla traccia dove è presente il settore che ci interessa. Dopodiché bisogna aspettare il tempo di rotazione in maniera tale che il settore di interesse vada a finire esattamente sotto la testina e dopo che questo è avvenuto c'è la caduta della testina e dunque avviene la lettura.

La cosa che pesa di più quando bisogna andare a cercare un dato è il movimento del braccio lungo il raggio.

Altre caratteristiche sono:

- I piatti del disco ruotano da 60 a 200 volte al secondo.
- **Velocità di trasferimento**: è la velocità con cui i dati vengono trasferiti dal disco al computer.
- **Tempo di posizionamento (o tempo di accesso)**: è il tempo per muovere la testina sul settore desiderato (tempo di ricerca + latenza di rotazione).

In generale il disco viene visto come un grosso array monodimensionale di blocchi, malgrado la sua struttura. È importante capire come vengono considerati i settori nelle tracce. Ci sono due tipi di politiche a riguardo: ci sono i dispositivi **CVL** (constant linear velocity) in cui la densità dei bit per traccia è uniforme, cioè le tracce esterne contengono più settori. Poi ci sono i dispositivi **CAV** (constant angular velocity) in cui la densità di bit decresce dalle tracce interne verso quelle esterne.



Lo spazio all'interno di un settore è gestito nel seguente modo: inizialmente c'è una serie di informazioni relative al settore, per esempio il numero del settore in analisi. Successivamente c'è la zona dati che viene utilizzata per immagazzinare dati. Una parte importante nel settore è quella che in figura è rappresentata come **ECC** che è l'Error Correcting Code che serve a verificare che la zona dati interessata dal settore non abbia subito errori o problemi.

Quando si memorizzano dei dati all'interno del settori, viene calcolato un numero che viene depositato all'interno dell'ECC. Questo serve per esempio quando all'interno di questo blocco vengono effettuate operazioni di lettura/scrittura. Viene ricalcolato questo numeretto e viene confrontato con quello precedentemente calcolato: se sono diversi sorge il dubbio che c'è stata qualche anomalia e si avverte il SO. Questo numeretto è una sorta di codice di sicurezza.

La prima cosa che il SO fa con un disco è quella di formattarlo (crea i settori e le numerazioni etc…). Dopo la creazione delle partizioni occorre formattarle logicamente. Per esempio, il sistema operativo ha bisogno di registrare su disco le partizioni e le corrispondenti directory, nonché le proprie strutture dati per la gestione dello spazio non allocato (creazione di un file system).

Per la gestione del disco è importante considerare la gestione dei blocchi difettosi. Quello che può succedere è che man mano che si utilizza il disco, si possono identificare una serie di blocchi che sono difettosi (per difettoso s'intende un problema di natura fisica). Questi blocchi naturalmente non possono essere utilizzati. In questi casi ci sono varie tecniche:

- **Sector sparing** (accantonamento dei settori): il controllore accantona i blocchi difettosi e li sostituisce con dei blocchi di riserva. Quando si formatta un disco, quello che succede è che non tutti i blocchi vengono messi a servizio dei dati. Una parte viene accantonata come settore di riserva. Laddove ci sono poi dei blocchi difettosi, si usano poi questi blocchi di riserva. Se si fa richiesta di scrittura ad un blocco difettoso, il controllore traduce la richiesta al blocco di riserva associato a quello difettoso.
- **Sector slipping** (traslazione dei settori): non c'è un vero e proprio settore di riserva contenente tutti i blocchi di riserva, bensì questi sono posizionati in posizioni casuali del disco. In questo caso tutti i settori compresi tra il settore dannegiato e quello di riserva immediatamente successivo, i dati, vengono spostati avanti di un settore.

4.1 SCHEDULING DEL DISCO

Vediamo ora come il driver del disco gestisce le richieste che arrivano al disco stesso. Supponiamo che nella macchina stiano girando una serie di processi e si stanno usando vari file. Al driver del SO possono arrivate ad un certo istante una serie di richieste di blocchi (ad esempio, il processo 1 richiede il blocco 10, il processo 2 il blocco 15 e così via). Il driver del SO deve gestire la richiesta di questi blocchi. Lo scheduling rappresenta una sorta di regola che deve essere rispettata dal driver del SO che gli consente di scegliere secondo un certo criterio quale processo deve essere servito per prima e quindi quale blocco mettere a disposizione per primo.

Si potrebbe pensare banalmente l'idea "servo il primo che richiede", ma questa non è detta che sia la politica migliore. Per capire qual è l'algoritmo di scheduling più giusto, si deve capire qual è atto pesa di più in termini di tempo per accedere ai vari blocchi.

Analizziamo dunque il **tempo di posizionamento** che ha due componenti principali:

- **Tempo di ricerca** (seek time): il tempo che impiega il braccio del disco a muovere le testine fino al cilindro contenente il settore desiderato. Questa componente pesa molto in termini di tempo.
- **Latenza di rotazione**: è il tempo aggiuntivo speso in attesa che il disco faccia ruotare il settore desiderato sotto la testina. Questa componente pesa poco in termini di tempo.

L'**ampiezza di banda** del disco è data dal numero totale di byte trasferiti diviso per il tempo totale che intercorre fra la richiesta di servizio e il completamento dell'ultimo trasferimento.

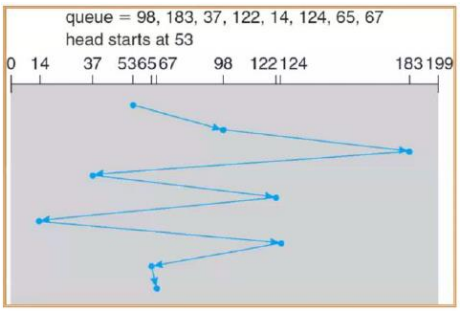
Quando si analizzano gli algoritmi di scheduling del disco si tiene conto quasi esclusivamente del seek time. L'obiettivo è minimizzarlo.

Supponiamo di avere un disco in cui le tracce sono numerate da 0 a 199 e supponiamo si presenti la seguente coda di richiesta sulle tracce: 98, 153, 37, 122, 14, 124, 65, 67.

Sappiamo inoltre che la testina è posizionata sul cilindro 53. Vediamo i vari algoritmi che ci permettono di gestire queste tracce:

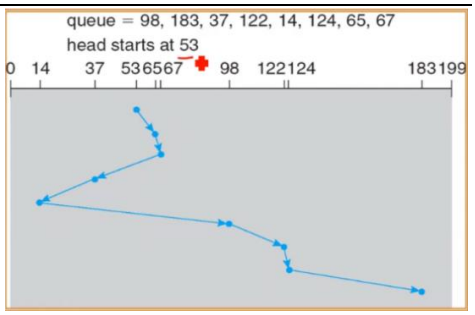
4.1.0 FCFS (FIRST COME FIRST SERVED)

L'idea è quella di servirli nell'ordine in cui le richieste sono arrivate.
Di conseguenza viene servito prima il 98, poi il 183, poi il 37 e così via.
Dal punto di vista dell'immagine vediamo come inizialmente, da traccia, la testina sta sulla posizione 53, successivamente si sposta verso il 98. Da 98 a 183 e così via.
Dovendo valutare tutti questi movimenti della testina quello che si fa è vedere quanti spostamenti si sono accumulati. Per esempio, mi pongo la seguente domanda: quanto tempo ho perso per andare da 53 a 98? 98-53= 45. Da 98 a 183, 85.
Continuando su questa idea alla fine scopriremo che il movimento totale della testina è stato di 640 cilindri. Si può fare di meglio?



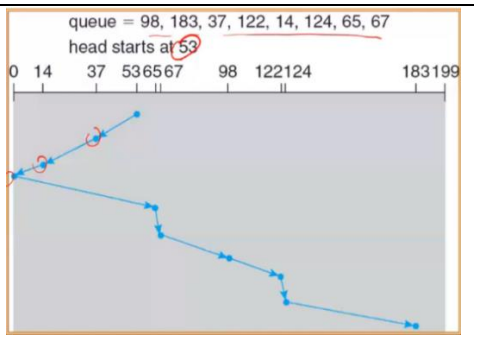
4.1.1 SSTF (SHORTEST SEEK TIME FIRST)

A partire dalla posizione attuale del cilindro, ci si va a spostare nei posti più vicini alla posizione attuale del cilindro, tra tutte le richieste che devono essere soddisfatte. Tenendo conto che la posizione iniziale del cilindro è 53, guardo tutte le richieste che ho nella queue e mi rendo conto che la richiesta più vicina alla posizione 53 è la 65. Mi sposto quindi alla posizione 65, poi vedo la richiesta che è più vicina alla posizione attuale della testina ed è la 67, mi sposto lì e così via.
Se consideriamo il tempo, il movimento totale della testina è 263 che, confrontato con il 640 di prima, è molto meglio. Attenzione però al fatto che questo tipo di schedulazione può causare attesa indefinita di richieste (**starvation**). Supponiamo che arrivino una serie di richieste che oscillano tra 60 e 70. Se c'è una richiesta alla posizione 180, questa non verrà mai servita.



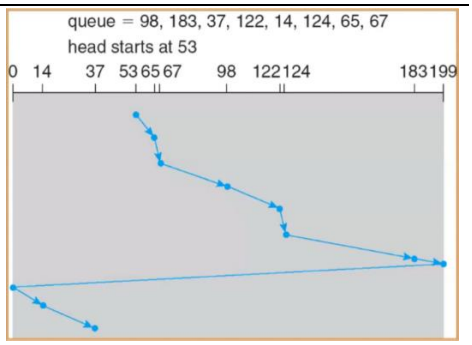
4.1.2 SCAN

Per garantire che tutte le richieste possano sempre essere servite, si muove il braccio del disco da un estremo all'altro. Per esempio, supponendo che la posizione iniziale della testina sia 53, e che il movimento della testina sia verso lo 0. Allora nel tragitto verso lo 0 vengono servite man mano tutte le varie richieste (37-14). Arrivato alla posizione 0 verranno servite in ordine crescente. In questo modo tutte le richieste sono potenzialmente raggiungibili, evitando chiaramente lo **starvation**.
In questo caso il movimento totale della testina è 208 cilindri.
Ma una volta soddisfatta la richiesta alla posizione 14, che senso ha dover andare anche alla posizione 0? Da 14 si potrebbe pensare di andare a soddisfare direttamente la richiesta alla posizione 65. Un altro difetto è che dalla posizione 0 alla posizione 65 sicuramente non ci saranno altre richieste che verranno servite e dunque questo è tutto tempo che viene sprecato inutilmente.



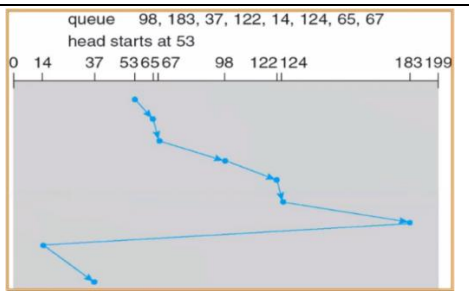
4.1.3 C-SCAN

Il braccio si muove da un capo all'altro del disco, servendo le richieste lungo il percorso quando raggiunge l'altro capo, ritorna direttamente all'inizio del disco (gestione mediante lista circolare) senza servire alcuna richiesta durante il ritorno. Le tracce vengono sempre viste come all'interno di una lista circolare e vengono servite **sempre in una direzione**. Dunque, la posizione della testina parte da 53 e si muove verso destra. Vengono servite tutte le richieste fino a 183. Da 183 si sposta a 199, dopodiché avviene un passaggio immediato da 199 a 0 (proprio perché è visto come una lista circolare). Da 0 ricomincia sempre in senso crescente e serve le ultime richieste che aveva.
Anche in questo algoritmo il problema è il seguente: una volta servita la richiesta 183, che senso ha dovermi spostare fino a 199, e da 199 andare a 0? Non potrei pensare di spostarmi da 183 a 14 direttamente?



4.1.4 C-LOOK

C-LOOK risolve il problema presentato qui sopra: arrivato alla richiesta 183, successivamente si riparte dalla prima richiesta più piccola e cioè 14 e si va in senso crescente fino a terminare le richieste.



4.2 UNITA' A STATO SOLIDO

All'interno delle SSD (Solid State Drive) non sono presenti dei dischi, ma si basano su memoria flash. Non è richiesta alcuna componente meccanica o magnetica ed è un dispositivo di memoria secondaria che ha una struttura totalmente differente rispetto a quelle viste precedentemente. L'unico difetto delle SSD è il numero delle possibili scritture che possono essere fatte (difetto relativo in quanto sono nell'ordine dei milioni).
Caratteristiche delle SSD sono:

- Meno soggetti a danni.
- Più silenziosi (non c'è movimento meccanico).
- Più veloci (non c'è seek).
- Non necessitano di defrag (per ridurre il tempo di seek).