Scheduling del Disco

L'obbiettivo dello scheduling del disco è di **massimizzare il numero di richieste** soddisfatte in una unità di tempo (throughput) ed inoltre **minimizzare il tempo medio di accesso**.

Abbiamo isolato le componenti principali che costituiscono il ritardo nel tempo di accesso al disco :

- tempo di posizionamento (seek-time) : tempo necessario al braccio del disco rigido per posizionarsi sull'area del disco in cui si trova il dato richiesto
- latenza di rotazione : tempo necessario affinché il settore del disco che contiene il dato richiesto si sposti sotto la testina di lettura/scrittura

Ottimizzazione del seek time

In un sistema, soprattutto se multiprogrammato, si vengono a creare varie richieste di I/O su disco che però, tipicamente, possono essere inviate al controller del disco solo una alla volta. Si crea quindi una **coda di richieste** pendenti;
Il SO può adottare varie politiche di selezione della prossima richiesta da mandare, si parla quindi di scheduling di selezione della richiesta da inviare.

First come first served (FCFS)

Il SO cerca di ottimizzare il costo maggiore, ovvero quello del costo di posizionamento o seek-time. L'idea è quello di fornire un ordine differenze da quello della coda, ovvero FCFS. Si cerca di minimizzare il costo relativo alla singola richiesta per ottenere un vantaggio su un intero set di richieste relative ad un settore del disco. Si considera solo il cilindro coinvolto dall'operazione e scartiamo informazioni marginali come la tipologia dell'operazione (lettura e scrittura hanno lo stesso costo nel caso del disco meccanico).

Il primo movimento va contestualizzato alla posizione della testina prima dell'esecuzione della prima richiesta. Si valuta anche il numero e l'ampiezza dei movimenti effettuati dalla testina (in termini di cilindri). L'implementazione di questo algoritmo è semplice e gestisce in modo equo le richieste, in quanto non favorisce nessuna richiesta. Tuttavia è poco efficiente.

Shortest Seek Time First (SSTF)

Si pensa di applicare lo scheduling per brevità al caso di scheduling del disco, in particolare si cerca di eseguire la richiesta che comporta uno spostamento minore della testina ovvero **la richiesta più vicina** nei confronti della posizione iniziale della testina.

L'algoritmo può diventare non equo nella misura in cui si favorisca l'esecuzione di richieste localizzate ad una stessa zona del disco, mantenendo i movimenti della testina in una stessa zona. Questo sfavorisce molto un sottogruppo di richieste che fanno riferimento ad un altra zona del disco e queste richieste si vedranno continuamente scavalcate (**starvation**).

Scheduling per scansione (algoritmo dell'ascensore)

In questo algoritmo lo stato attuale della testina ha un verso iniziale (crescente o decrescente). Si scheduleranno le richieste che fanno riferimento facendo una scansione e selezionando solo quelle nel verso della testina, ad esempio se il verso è crescente si schedulano solo quelle con numero crescente e al termine di queste cambierà il verso della testina e si procederà schedulando tutte le richieste da destra verso sinistra in senso decrescente.

Questo algoritmo garantisce un tempo di attesta massimo per le richieste prima che vengano eseguite, fornendo una sorta di equità nella gestione delle richieste. Risulta noto a priori un upper bound pari al doppio del numero di cilindri disponibili nel caso peggiore (worst case) che si verifica quando una nuova richiesta viene inserita dopo che la testina ha appena scandito quella posizione ed infatti dovrà ri-aspettare che la testina ripassi da quella posizione (deve fare andata e ritorno). In genere questa soluzione fornisce garanzie ed è considerata migliore dello scheduling per brevità.

Scheduling per scansione circolare

Una variante prevede che al termine dello scorrimento della coda non si ritorni indietro in maniera inversa a come si era proceduto ma si riprende la gestione delle richieste rientrando dall'estremo opposto della coda.
Quindi verranno ignorate richieste pendendenti sulle posizioni "saltate" ed in linea di principio questa scelta migliora i tempio di attesa, questo perchè ritornando indietro come nell'algoritmo dell'ascensore allora si scheduleranno altre richieste relative alla stessa zona dove si è appena passati, andando a favorire di nuovo richieste localizzate ad una stessa zona del disco facendo avanzare i tempi medi di attesa. Quando si ha un sistema ad alto carico di richieste risulta efficiente la variante circolare dell'algoritmo .

Sistemi RAID

Un altro modo per aumentare le prestazioni è sfruttare il parallelismo anche per l'I/O su disco: in particolare si utilizzano diversi dischi con prestazioni modeste piuttosto che un unico disco con prestazioni eccellenti, in modo da abbassare anche i costi economici. I diversi dischi risultano indipendenti tra loro ed è importante avere nell'intero set solo dischi con prestazioni simili tra loro se non addirittura identici. Si suddividono i dati relativi ad una unità logica (un file, in generale un volume) sul seti di dischi che costituiscono l'unità di storage, la cui capienza totale è data dalla somma delle capienze dei vari dischi. La suddivisione in più unità diviene trasparente all'utilizzatore in quanto di interfaccerà con l'astrazione di un unico disco RAID.

- **Raid-0 (striping)**

Ragionando su un singolo disco magari molto grande e si supponga di avere quattro dischi di dimensione minore. Si pensa di separare il disco, mappandolo in più parti detti **strip**. I vari strip verrano memorizzati secondo modalità Round Robin spalmandoli sulle varie unità di memorizzazione. Il vantaggio nel distribuire il disco grande in diversi dischi è dato dal fatto che il carico di richieste relative al file sarà gestito in maniera parallela da tutti i dischi, i quali si fanno carico di un gruppo di richieste relative agli strip del file memorizzati. Si ottiene uno speed-up che incrementa il throughput di richieste di un fattore 4 nel caso si abbiano 4 dischi. Nel caso peggiore le richieste riguarderanno tutti e soli gli strip nello stesso disco, questo caso vedrà la gestione equivalente al caso di un solo disco.

Il problema è che facendo striping su più unità si aumenta la probabilità che il disco logico RAID si guasti, infatti con 4 dischi la probabilità che il volume logico si guasti è data dalla probabilità che si guasti ognuno dei 4 dischi che lo compongono, infatti quadruplica.

- **RAID-1 (mirroring)**

Si cerca di attenzionare la resistenza al guasto del sistema di storage (migliore fault tollerance) e non della sola efficienza nella gestione delle richieste. Si introduce una ridondanza data dalla duplicazione dei dischi fisici in altrettanti dischi di mirroring che serviranno per ripristinare lo stato dell'intero sistema di storage. Se un disco si guastasse lo si ripristina e in maniera automatica il sistema stesso lo ripopola con i dati che erano presenti nella copia del disco guastato. Si ha la stessa prestazione di RAID-0 ma si paga un costo dovuto al dover inserire ulteriori disci per fare il mirroring. Anche le prestazioni in lettura non si vedono eccessivamente penalizzate da questo modello .

- **RAID-2 (striping a livello di bit con ECC codici di correzione degli errori)**

L'esigenza di risparmiare sul numero di dischi fisici porta all'introduzione di uno striping a livello di bit e non di dischi. Si considerano word da 4 bit nel volume logico e si introduce una ridondanza di 3 bit tramite il codice di Hamming e poi si mappano i 7 bit nei 7 dischi reali. Questo consente di perdere un solo bit per tutte le word se si dovesse guastare un disco, ad esempio se si guasta il disco 3 perdo il terzo bit di tutte le word, ma la perdita di un solo bit si riesce a ripristinare facilmente tramite il codice di Hamming stesso.

Questo permette di risparmiare sull'ottavo disco in quando ne sono presenti solo sette, tuttavia affinchè funzioni questo meccanismo si necessita di avere una sincronizzazione nel lavoro di tutti e sette i dischi. Per la ricostruzione di un blocco si sfruttano le informazioni negli altri dischi e se uno di questi non è sincronizzato con gli altri il sistema non funziona.

- **RAID-3 (strip a livello di bit con bit di parità)**

Aggiungere tanti bit di ridondanza necessari alla correzione degli errori risulta poco efficiente e si può pensare di introdurre invece una ridondanza minimale che permette la sola rilevazione degli errori, ad esempio un bit di parità. Si introduce un solo disco in più per mantenere l'informazione dei soli bit di parità necessari alla ricostruzione della corretta sequenza di bit.

Quando si verifica un guasto al terzo disco si deduce banalmente che la sequenza ha riscontrato un errore nel terzo bit della sequenza da quattro, ed essendo facile la rilevazione dell'esatto bit si può facilmente **correggere** il valore del bit stesso. La capienza totale del set di storage è data dalla somma delle capienze dei primi quattro dischi.

- **RAID-4 (striping a livello di blocchi con XOR sull'ultimo disco)**

Si ritorna all'idea di RAID-0 ovvero allo striping ma estendendo l'idea del bit di parità, che viene rimpiazzato da uno XOR logico. Anche qui viene introdotto un solo disco di ridondanza che contiene il valore di XOR del valore di tutti gli altri bit allineati negli altri dischi. Se dovesse guastarsi il disco 2 si ri-popola facendo lo XOR di tutti gli altri bit rimanenti.
La miglioria rispetto a RAID-3 risiede nel fatto che lavorando per strip e non per bit non necessita di sincronizzazione per la ricostruzione dell'unico bit perso.

- Le scritture nel caso di RAID-0 vengono spalmate esattamente come le letture, ovvero con accessi a diversi dischi.
- Le scritture nel caso di RAID-1 vengono replicate sul disco e riportate sul relativo disco di mirroring ma in modo parallelo e quindi non necessita del doppio del tempo.
- Le scritture nel caso di RAID-4 invece risultano più costose rispetto agli altri modelli : infatti si paga un accesso in scrittura sia sul disco che contiene lo strip modificato e sia sul disco degli XOR di ridondanza. Tuttavia prima di effettuare una scrittura sul disco di ridondanza bisogna leggere il valore di tutti gli altri e calcolare il nuovo valore prima di scriverlo. Si può invece apportare una miglioria andando a sostituire il valore nel disco di ridondanza con uno XOR tra il vecchio valore che era presente nel disco di ridondanza stesso, il vecchio ed il nuovo valore del disco.

Tuttavia si presenta un problema di prestazioni dovuta al fatto che un disco (quello di ridondanza) lavora a seguito di qualunque scrittura in qualunque altro disco.

- **RAID-5 (RAID-4 con informazioni di parità distribuite)**

Il problema di RAID-4 viene risolto distribuendo su tutti gli altri dischi le informazioni presenti sul bit di ridondanza. Come risultato di questa sistemazione si ha che vengono preservate le stesse proprietà di RAID-4 ma si distribuisce il carico sui dischi che riportano informazioni non ridondanti.

SSD (Solid State Disk)

Gli SSD sono molto più apprezzati dei dischi elettromeccanici per varie motivazioni. Un primo motivo è legato al fatto che le letture sono molto più veloci delle scritture. Questo è dovuto all'ingegneria dell'SSD stesso, che implica che prima di sovrascrivere un blocco, il suo contenuto si deve prima cancellare per poter essere riscritto. Un lato negativo è dettato dal fatto che dopo un certo numero di sovrascritture il blocco smette di funzionare e non riesce ad immagazzinare altre informazioni.

I blocchi sono composti da unità di allocazione e le operazioni di lettura e scrittura sono basate su pagine, per tanto un blocco può risultare pieno per metà. L'inconveniente è dettato dal fatto che per la sostituzione di una sola unità di allocazione si necessita di sovrascrivere l'intero blocco. Ogni pagina è pilotata da un controller nel device stesso. Una componente detta **Flash Translation Layer** si occupa di mappare il contenuto di pagine del disco nelle unità che costituiscono i blocchi.

SSD (Solid State Disk)

Gli SSD sono molto più apprezzati dei dischi elettromeccanici per varie motivazioni. Un primo motivo è legato al fatto che le letture sono molto più veloci delle scritture. Questo è dovuto all'ingegneria dell'SSD stesso, che implica che prima di sovrascrivere un blocco, il suo contenuto si deve prima cancellare per poter essere riscritto. Un lato negativo è dettato dal fatto che dopo un certo numero di sovrascritture il blocco smette di funzionare e non riesce ad immagazzinare altre informazioni.

I blocchi sono composti da unità di allocazione e le operazioni di lettura e scrittura sono basate su pagine, per tanto un blocco può risultare pieno per metà. L'inconveniente è dettato dal fatto che per la sostituzione di una sola unità di allocazione si necessita di sovrascrivere l'intero blocco. Ogni pagina è pilotata da un controller nel device stesso. Una componente detta **Flash Translation Layer** si occupa di mappare il contenuto di pagine del disco nelle unità che costituiscono i blocchi.

Garbage collection

Quando si mappano le pagine sui blocchi lo si fa tenendo conto di quali unità sono occupate e in quel caso vengono marchiate come non valide e la nuova informazione verrà inserita in un altra unità libera.
Quando un blocco è pieno si spostano le pagine valide di quel blocco in un altro che è libero e si svuota completamente il blocco che era saturo di dati e non poteva più essere usato per la scrittura. Svuotando l'intero blocco lo si avvicina all'invecchiamento e l'idea è quella di mantenere un'usura omogenea dei blocchi.

TRIM

Quando un blocco di memoria SSD viene cancellato, i dati vengono marcati come "non validi" ma rimangono fisicamente presenti sulla memoria flash finché non vengono sovrascritti con nuovi dati. Quando un blocco è quasi pieno, appare al controller dell'SSD come tale, ma può ancora apparire al sistema operativo come disponibile, poiché il sistema operativo non è a conoscenza della gestione interna dei blocchi dell'SSD. TRIM consente al sistema operativo di comunicare al controller dell'SSD quali blocchi non sono più in uso e possono essere cancellati in anticipo, migliorando le prestazioni dell'unità SSD.

SSD vs Disco Magnetico

A differenza dei dischi rigidi tradizionali, gli SSD non hanno parti mobili, il che significa che non devono attendere il tempo di posizionamento (seek time) e la latenza di rotazione per accedere ai dati, il che li rende molto più veloci dei dischi rigidi. Inoltre, gli SSD sono generalmente più affidabili dei dischi rigidi tradizionali, poiché non esistono parti meccaniche che possono guastarsi.