

Gestione e ottimizzazione del file system

Gestione dello spazio su disco

I file sono memorizzati sul disco, ci sono due maniere principali:

- **Allocazione contigua** --> Se la dimensione aumenta, allora lo dovremmo spostare da una parte all'altra del disco.
- **Blocchi non contigui** --> File spezzettati in blocchi di dimensioni fisse consentendo maggiore flessibilità e migliore utilizzo dello spazio.

Dimensione dei blocchi

La comune dimensione è di 4KB per blocco. Nasce da un compromesso:

- Blocchi troppo grandi -> Spreco di spazio da parte di file piccoli.
- Blocchi troppo piccoli -> Spreco di tempo nella ricerca perché spezzettiamo i file in più blocchi.

La scelta ottimale bilancia il tempo di trasferimento e l'efficienza dello spazio.

Implicazioni per tipi diversi di memoria

- **Dischi magnetici:** Scelta dei blocchi influenzata dal tempo di ricerca e dal ritardo di rotazione. Se aumentiamo la dimensione dei blocchi, incrementiamo la velocità di trasferimento, ma riduciamo l'efficienza dello spazio.
- **Memoria Flash:** Può avere sprechi di spazio sia con blocchi grandi che piccoli a causa delle dimensioni fisse delle pagine flash.

Tendenze attuali

Con l'incremento delle capacità dei dischi potrebbe essere vantaggioso considerare blocchi più grandi per accettare una minore efficienza dello spazio in cambio di prestazioni migliori.

Adesso, come facciamo a tenere traccia dei blocchi liberi?

Abbiamo due metodi, tramite:

- **Lista concatenata di blocchi del disco** -> Ogni blocco della lista contiene **tanti numeri di blocchi del disco liberi quanto possibile.**
- **Bitmap** -> Un disco con n blocchi richiede una bitmap ad n bit, i blocchi liberi sono indicati con uno 0.

Lo schema della lista concatenata richiede meno spazio della bitmap solo se il **disco è quasi pieno.**

Possiamo migliorare l'approccio della lista concatenata tramite delle modifiche.

Invece di tracciare dei blocchi singoli, **tracciamo serie di blocchi consecutivi**, ogni blocco è assegnato un **conteggio** (8, 16, 32 bit) che rappresenta il numero di **blocchi liberi consecutivi.**

⚠ Problema

E' molto efficiente per dischi vuoti, ma poco per quelli frammentati.

Il problema principale è che ci sono molti algoritmi e strutture dati che si possono usare, però la scelta migliore richiede dati che i progettisti non possono avere finché il sistema non sarà rilasciato.

Tornando al metodo della lista dei blocchi liberi, adesso dobbiamo:

- Mantenere in memoria solo un **blocco di puntatori.**

Creazione di un file --> I blocchi necessari vengono presi dal blocco dei puntatori.

Eliminazione file -> I blocchi sono liberati e aggiunti al blocco dei puntatori.

Quando i puntatori finiscono viene letto un nuovo blocco di puntatori dal disco.

Questo metodo **evita I/O** su disco inutili mantenendo una lista di blocchi interi direttamente accessibili in memoria.

Per impedire che gli utenti occupino troppo spazio, l'amministratore può scegliere di assegnare delle **quote del disco**, cioè un limite massimo di file e blocchi, descriviamone il meccanismo:

1. Utente apre un file -> attributi ed indirizzi vengono localizzati e posti in una **tabella dei file aperti** nella memoria principale, incrementi del file sono contabilizzati nella quota del proprietario.
2. Una seconda tabella contiene il **record delle quote di ogni utente** che ha un file attualmente aperto, i record sono aggiornati ad ogni chiusura di file.

La tabella dei file aperti a un puntatore al record dell'utente

Esistono diversi tipi di limiti:


- **Soft** --> Possono essere momentaneamente superati.
- **Hard** --> Non possono essere mai superati.

Violazioni possono portare a restrizione dell'accesso.


Backup del file system e Punti chiave di un Backup

Distruzione del file system è peggiore della distruzione di un computer.

Perdere il file system non è buono, perché **recuperare tutte le informazioni sarà difficile**. Inoltre la perdita di alcuni dati potrebbe risultare drammatica.

 **Il file system non può offrire alcuna protezione contro la distruzione fisica dei dispositivi, però può aiutare a proteggere le informazioni.**

La soluzione per evitare questi problemi è quello di fare copie di salvataggio, chiamati **backup**.

 **I backup si fanno per gestire uno di due potenziali problemi:**

1. Ripristino di dati per a causa di un disastro;
2. Ripristino di dati persi a causa di un errore dell'utente.

Primo caso --> Crash del disco, Incendio, Allagamento --> Eventi **rari**.

Secondo caso --> Eliminazione di un file da parte dell'utente --> Molto **comune**.


Windows

Quando eliminiamo un file su Windows, lo stiamo soltanto spostando nella directory cestino.

Un backup impiega molto tempo e occupa molto spazio, bisogna quindi saperlo fare in maniera efficiente.

Per esempio **non è necessario fare il backup di tutto** il file system, poiché alcuni file sono di facile installazione o perché alcuni file non hanno senso di essere copiati, poiché pericolosi se il programma di backup provasse a leggerli (Es: file nella directory /dev di UNIX).

Inoltre è uno **spreco rifare il backup di file che non sono stati cambiati** dall'esecuzione del primo backup.

 **Backup incrementale**

Fare il backup di file modificati settimanalmente, però il ripristino è complicato.

Considerando che stiamo memorizzando molti dati **potrebbe essere utile comprimere i file.**

⚠ **Bisogna prestare molta attenzione del backup dei dati, un solo punto difettoso potrebbe compromettere il backup.**

E' **difficile eseguire il backup di un file system attivo**, se durante il backup avvengono operazioni di cancellazione, modifica di file il risultato del backup potrebbe risultare incoerente.

✍ **Esistono algoritmi di backup che fanno uno snapshot della situazione corrente del file system in modo tale da ricopiare soltanto quello che visto nello snapshot.**

L'ultima cosa di cui bisogna parlare è di **sicurezza**: I backup devono essere custoditi in zone che si trovano al di fuori del luogo dove risiedono i computer.

Come è organizzato EXT2

Le componenti chiave del file system EXT2 sono:

- **Superblocco** -> Contiene informazioni sul **layout, numeri di I-node, blocchi del disco.**
- **Descrittore del gruppo** -> Contiene dettagli sui **blocchi liberi, I-node**, e posizioni delle bitmap.
- **Bitmap** -> Traccia i **blocchi liberi e gli I-node liberi.** (Design MINIX 1).

✍ Inode

E' una struttura dati che archivia e descrive attributi base su file, directory o qualsiasi altro oggetto.

Contengono informazioni di contabilità e indirizzi dei blocchi di dati.

✍ Blocchi di dati

Archiviano i file e le directory, non necessariamente contigui sul disco.

Gli I-node delle directory sono distribuiti tra i **gruppi di blocchi del disco**. Si cerca di posizionare file nella stessa area di blocchi della directory genitore per minimizzare la frammentazione.

Le bitmap vengono usate per determinare rapidamente le aree libere dove allocare nuovi dati del file system.

In particolare Ext2 **prealloca in anticipo otto blocchi** aggiunti per ogni nuovo file in modo tale da ridurre la frammentazione dovuta a scritture successive.

Questa strategia bilancia il carico del file system e migliora le prestazioni riducendo la frammentazione.

Per accedere ai file vengono usate delle **chiamate di sistema** (Es: open), mentre per quanto riguarda le ricerche nelle directory queste sono **lineari**, ma ottimizzate da una **cache** delle directory recentemente accedute.

Aprire un file --> Percorso analizzato per trovare l-node della directory e poi l-node sul file.

In Ext3/Ext4 si gestisce meglio la frammentazione, riducendo la necessità di frammentazione manuale tramite la **preallocazione di blocchi**.

Funzionamento

Scrittura del file, si prealloca un gruppo di blocchi contigui, riducendo la frammentazione per file in espansione.

Performance del file system

Impieghiamo **molto più tempo per accedere al disco** che alla memoria, per questo sono stati progettati diversi file system che migliorano le prestazioni.

Tempo di accesso

Il tempo di accesso è maggiore a causa del tempo di ricerca della traccia e dell'attesa del posizionamento della testina di lettura.

Vogliamo quindi ridurre al minimo:

- I numeri di accesso al disco.
- Il tempo di ricerca del dato su disco.
- L'utilizzo dello spazio.

Block cache -> In questo caso sono una **raccolta di blocchi** appartenenti logicamente al disco, ma che sono tenuti in memoria per ragioni di prestazione.

Per gestire la cache possiamo usare diversi algoritmi:

- Comune --> Controlla le richieste di lettura e verifica se il blocco è nella cache. Se non ci sta viene preso dal disco e copiato all'interno della cache.

Deframmentazione

Con il tempo i dischi si frammentano, i file sparsi portano a prestazioni inferiori, la deframmentazione **riorganizza i file sparsi in modo tale da essere contigui**.

Windows fornisce lo strumento defrag. Utile per HDD. Sconsigliato per SSD.

Per ottimizzare la ricerca di un blocco nella cache usiamo delle funzione **hash**.

Quando invece un blocco deve essere sostituito possiamo usare i classici algoritmi di paginazione come:

- **FIFO**
- **Seconda Chance**
- **LRU**

La struttura della cache permette l'implementazione corretta del LRU, ma ci accorgiamo che non è una cosa gradita.

Ci sta un intoppo che ha a che fare con i **crash e la coerenza del file system**, infatti se un blocco critico viene letto nella cache e modificato, ma non riscritto nel disco, un crash del file system lo lascerebbe in uno stato incoerente.

Dobbiamo creare un nuovo schema LRU che tenga conto di due cose:

1. E' probabile che presto il blocco servirà ancora?
2. Il blocco è fondamentale per la coerenza del sistema?

Se il blocco è critico allora **avviene una scrittura immediata** che riduce la possibilità che un crash distrugga il file system.

 **Comunque i file non devono restare troppo tempo nella cache.**

In UNIX abbiamo una chiamata di sistema **sync** che ==forza la scrittura immediata ==su disco di tutti i blocchi modificati.

In Windows abbiamo la strategia **write through**, cioè facciamo una scrittura immediata per i blocchi modificati (Integrata con *FlushFileBuffers).

Un'altra strategia è quella di **combinare cache delle pagine e cache buffer** in modo tale da ottimizzare I/O e gestire la memoria efficacemente.

Utile soprattutto se sono supportati i file mappati in memoria.

Funzionamento

Trattiamo i blocchi dei file e le pagine del file in un'unica cache, in questo modo se il file è mappato in memoria non occupa un'area di memoria per le pagine, ma sfrutta il cache buffer del disco.

Compressione e deduplicazione

Oggi ci sono **molti dati** e i dispositivi di memorizzazione si riempiono molto velocemente. Troviamo un modo per evitare che si riempiano subito.

La tecnica principale usata è la **compressione**, fornita da diversi file system che comprimono automaticamente cartelle e file. Alcuni file system sono:

- **NTFS (Windows)**
- **Btrfs (Linux)**
- **ZFS (Altri)**

Algoritmi di compressione -> **Cercano sequenza di dati ripetuti**, codificabili in modo efficiente.

Alcuni file system eliminano la ridondanza in tutti i file, specialmente nei sistemi che memorizzano **molti dati da parte degli utenti**. (Cloud, Server, Backup).

Invece di salvare gli stessi dati usiamo la **deduplicazione** per eliminare i doppi.

Quando la procedura di deduplicazione rileva che due file contengono porzioni del tutto uguali, conserva un'unica copia fisica condivisa da entrambi i file.

Può essere eseguita:

- *inline* -> Hash per ogni **chunk** che sta per scrivere e lo confronta con gli hash di quelli esistenti. Se già esiste viene aggiunto un riferimento al chunk già esistente.
- *post-process* -> Scrive sempre i dati ed esegue hashing e confronti in background, quindi non rallenta le operazioni di elaborazione dei file.

Problema dei cassetti

Se due brani hanno lo stesso hash risulta un problema, ma molti accettano il rischio.

Deframmentazione del disco

Installazione del SO --> Programmi e file sono installati consecutivamente a partire dall'inizio del disco.

Con il tempo vengono creati ed eliminati quindi il disco si frammenta e, se creiamo un nuovo file i **blocchi usati potrebbero essere sparsi per il disco** il che rallenta le prestazioni.

Alcuni file, come il file di paginazione, il file di ibernazione e il log di journaling, non possono essere spostati poiché l'amministrazione necessaria per lo spostamento è molto onerosa e non ne vale la pena.

file system linux --> Si frammentano di meno.

Coerenza del file system

Se il sistema ha un crash prima di riscrivere i blocchi modificati, il sistema può rimanere in uno **stato incoerente**.

Utility usate

UNIX --> *fsck*

Windows --> *sfc*

Sono eseguite all'avvio dopo un crash.

Alcuni file sistemi, come quelli con **journaling** sono progettati per gestire autonomamente la maggior parte delle incoerenze e non necessitano di controlli esterni dopo un crash.

Eliminazione sicura dei file e cifratura del disco

Gli elementi fisici dei dischi magnetici o SSD possono essere sempre letti estraendo il dispositivo di memoria di massa e leggendolo da un'altra macchina.

Implicazioni

Se elimino un file da disco, questo **non è davvero eliminato**, posso leggere i blocchi grezzi.

Eliminare completamente dati dal disco non è semplice. Non basta sovrascrivere tutto, poiché, in alcuni dischi, i **dati memorizzati lasciano dei residui magnetici ==nelle zone vicine alle tracce vere e proprie. E' necessario cancellare anche ==aree adiacenti** come backup e cache.

Come eliminare un file

Fare diversi passaggi (da tre a sette) sul disco alternando zeri e numeri casuali, molte scritture mettono a dura prova gli SSD.

Inoltre negli SSD la mappatura dei blocchi flash è gestita dalla FTL il che rende la sovrascrittura meno prevedibile.

Un modo per rendere impossibile il recupero dei dati di un disco è la **cifratura completa di tutto il disco** e si può fare su tutti i SO.

La cifratura del disco è fornita anche dai dispositivi stessi, **SED** (SelfEncrypting Drives) che risparmiano alla CPU i calcoli crittografici, ma è stato scoperto che hanno delle **criticità** (2019).

Windows usa le capacità dei SED se ci sono, altrimenti usa una *chiave master del volume* nell'algoritmo **AES** (Advanced Encryption Standard).

Come ottenere la chiave master

Bisogna decifrare la chiave con la password utente, chiave di ripristino O TPM (Trusted Platform Module), cioè un cripto processore.