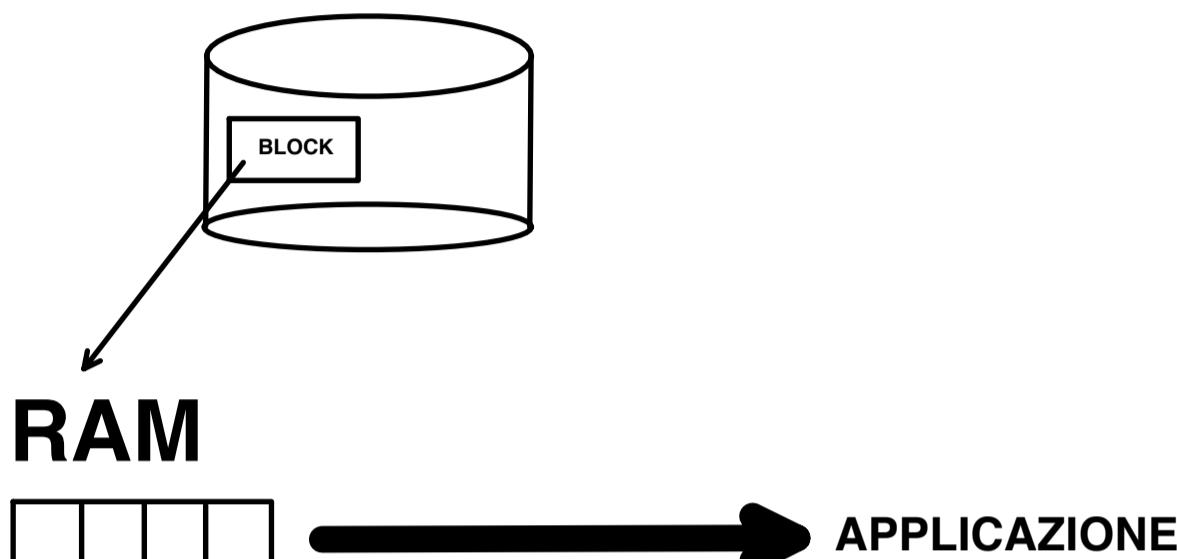


Cache dei dispositivi

È sempre vero che quando noi andiamo a lavorare su un FILE facciamo queste operazioni?

Supponiamo di avere un file F, e supponiamo di sapere tramite un record di sistema e un metodo di allocazione, che i dati del file sono all'interno di questo blocco. È sempre vero che quando un'applicazione ci chiede di leggere dei record di questo file e, passando il controllo al file system tramite una system call, questo file system per consegnare i record all'applicazione va a caricare ogni volta nuovamente il blocco in RAM, e una volta caricato prende i record richiesti e li riconsegna all'applicazione?

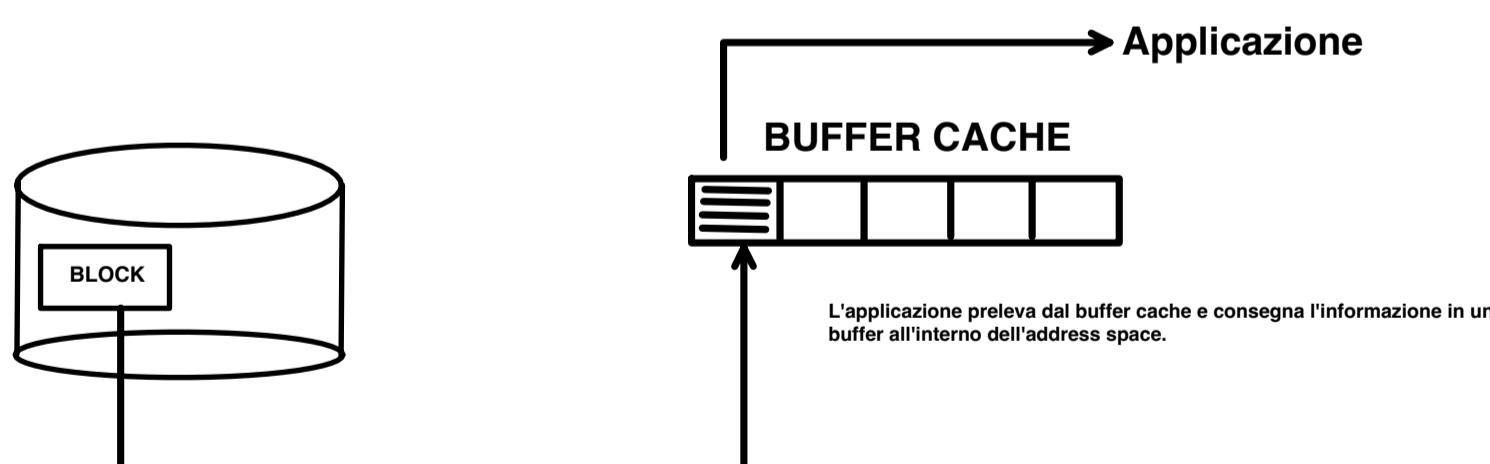


È sempre vero che un'applicazione che scrive dei nuovi record o aggiorna dei record che sono in qualche modo già presenti all'interno di questo file, passa il controllo al software del file system tramite una system call, e questo software del file system va ogni volta ad aggiornare i dati all'interno del blocco del dispositivo?

Quello che succede è che noi quando andiamo a lavorare all'interno di un file-system, in particolare nell'interazione che abbiamo tra il software del kernel e la struttura hardware dove noi manteniamo i dati di un file, quello che succede è che al livello del kernel abbiamo una zona di memoria che si chiama "BUFFER CACHE" DOVE MANTENIAMO I BLOCCHI DEL DISPOSITIVO.

Quindi quando alcuni record del blocco da leggere devono essere consegnati all'applicazione, leggiamo un blocco e lo carichiamo all'interno dell'elemento della BUFFER CACHE, NOI QUEL BLOCCO CE LO TENIAMO LÌ, all'interno del buffer cache.

Quindi questa area di memoria che il software del kernel mantiene per mantenere questi blocchi, è utile perché se un'applicazione legge nuovamente dei record che sono mantenuti all'interno di un blocco che io ho già caricato in questa zona, non ho più bisogno di interagire con il dispositivo di memoria di massa.



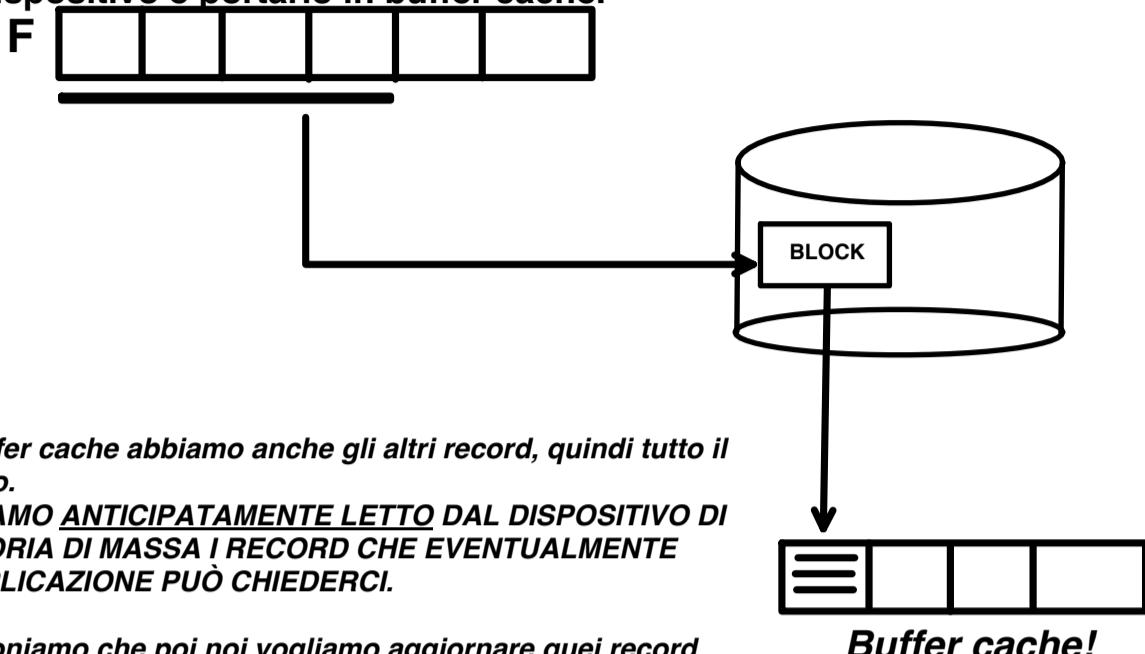
Poi i dati verranno riportati sulla copia originale, perché se così non fosse quando mandiamo in shutdown il sistema, perderemmo tutti i record.

Questo buffer cache ovviamente ha una taglia T limitata. Prima o poi tutti gli elementi del buffer cache saranno pieni e noi avremo necessità di caricare qualche altra cosa dal dispositivo di memoria di massa verso il buffer cache: ovviamente dobbiamo liberare uno di questi elementi per avere la possibilità di caricare un altro blocco nel momento in cui qualche applicazione mi richiede di lavorare in qualche specifico punto di qualche file, bisogna identificare un blocco vittima per eliminarlo e liberare spazio per il nuovo caricamento.

- il sistema operativo mantiene una regione di memoria che funge da buffer temporaneo (**buffer cache**) per i dati acceduti in I/O **e che siano riaccessibili**
- hit nel buffer cache evita interazione con gli hard-drive (diminuzione della latenza e del carico sugli hard-drive)

Possiamo rileggere gli stessi dati o riaggiornarli, possiamo anche riposizionarci e se abbiamo un metodo di accesso diretto possiamo riposizionarci sempre nella stessa zona del file.

Supponiamo che l'applicazione vuole leggere un certo record di un file F e questo record insieme ad altri è all'interno di un blocco di dispositivo di memoria di massa, e chiaramente per consegnare questo record noi dobbiamo prendere tutto il blocco del dispositivo e portarlo in buffer cache.



In buffer cache abbiamo anche gli altri record, quindi tutto il blocco.

ABBIAMO ANTICIPATAMENTE LETTO DAL DISPOSITIVO DI MEMORIA DI MASSA I RECORD CHE EVENTUALMENTE L'APPPLICAZIONE PUÒ CHIEDERCI.

Supponiamo che poi noi vogliamo aggiornare quei record che sono già rappresentati all'interno del buffer cache.

Quindi noi aggiorniamo il BUFFER CACHE ma non il dispositivo di memoria di massa, quest'ultimo viene aggiornato soltanto quando questi dati, dal buffer cache, vengono eliminati e riportati nel blocco della memoria di massa. E questa si chiama scrittura ritardata.

Questa è una cosa molto importante per cambiare il comportamento delle applicazioni da I/O Bound verso il CPU Bound. Nel momento in cui abbiamo un'applicazione che esegue in maniera frequente le operazioni di I/O, magari i dati su cui dobbiamo lavorare li abbiamo già nel buffer cache, quindi non dobbiamo bloccarci per attendere che un dispositivo ce li consegna, quindi riduciamo (avendo uno schema basato sul buffer cache) la probabilità di avere un blocco effettivo quando andiamo ad eseguire operazioni di I/O.

Quindi possiamo mantenere applicazioni che sono I/O intensive che eseguono operazioni intense sui file ad essere comunque CPU bound ed essere quindi molto più probabilmente pronte per andare in CPU.

Un'applicazione che entra spesso in stato di blocco si DICHIARA di essere un'applicazione I/O Bound.

Strategia di sostituzione dei blocchi

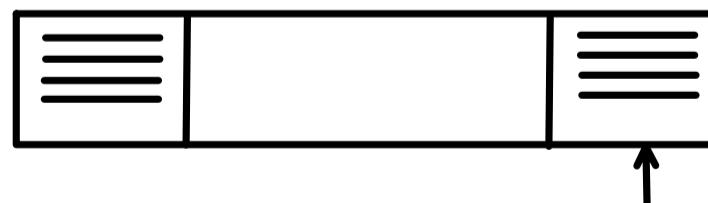
Per quanto riguarda la strategia di sostituzione dei blocchi che noi abbiamo all'interno di un buffer cache sono state proposte:

-Least-Recently-Used: Tra tutti questi blocchi che ho all'interno di questo buffer cache, mantengo traccia di qual è l'ultimo blocco del buffer cache che è stato utilizzato. A partire da questo mi ricordo che prima ho caricato quest'altro, poi ancor prima un altro e così via..!

In questo caso conosco qual è il blocco che più recentemente è stato caricato o più recentemente è stato utilizzato (perché posso aggiornare i blocchi nel BF anche quando avviene un'operazione su un blocco che era già presente all'interno del Buffer cache STESSO).

Supponiamo di avere un buffer cache con due elementi:

Io ho letto alcune informazioni e le ho caricate all'interno del buffer cache, poi successivamente ho letto altre informazioni da un file, caricando quindi all'interno del buffer cache i dati che sono presenti all'interno di un dispositivo di memoria di massa, e LEAST-RECENTLY-USED ci dice che questo è l'ultimo usato e poi abbiamo quello prima. Però se noi andiamo a scrivere su quello prima andiamo a cambiare la struttura di questa lista perché così diventa l'ultimo usato e il precedente diventa l'altro.



Se noi lavoriamo con applicazioni concorrenti, con CPU0 e CPU1, andiamo in I/O, sia su CPU0 E CPU1 dobbiamo andare a livello kernel a lavorare all'interno del buffer cache e quindi ad andare a riorganizzare la struttura della lista che ci dice qual è il least-recently-used all'interno di questo buffer cache.

-Least-Frequently-Used: Si mantiene un contatore di riferimenti per ogni blocco, che indica il NUMERO DEGLI ACCESSI da quando è stato caricato quel blocco all'interno del buffer cache. Quante volte sono stati letti o scritti quei dati del blocco da parte di un'applicazione.

Quindi quando dobbiamo eliminare un blocco all'interno della BUFFER CACHE, andiamo a vedere qual è quello più basso associato a ciascun blocco.

Questa cosa non ci piace quando l'elemento col contatore basso è stato caricato da poco mentre tanto tempo fa avevamo lavorato su un blocco con un contatore più alto. Quindi il BF è pieno, e abbiamo un blocco con un grande contatore su cui però abbiamo lavorato tanto tempo fa, quindi magari file CHE NON CI INTERESSANO PIÙ.

Mentre invece abbiamo un blocco su cui abbiamo incominciato a lavorare da poco con C' non cresciuto, ma vogliamo continuare a lavorare su queste informazioni.

PER OVIARE ABBIAMO UN BUFFER CACHE A SEZIONI MULTIPLE.

Buffer cache a due sezioni

C'è una zona del buffer cache che chiamiamo "nuova" e una zona che chiamiamo "vecchia".

Abbiamo un contatore associato a ciascun elemento (BLOCCO) di questo buffer cache, MA L'INCREMENTO del contatore viene effettuato quando l'elemento viene portato nella sezione NUOVA.

Quand'è che un blocco che non è presente in buffer cache viene portato in "sezione nuova"?

Quando questo viene acceduto dalle applicazioni! Le applicazioni cercano di accedere ai dati che sono associati ad un certo file - in particolare ai dati del blocco di memoria di massa - questo blocco non è presente all'interno del buffer cache e quindi lo carichiamo al suo interno, ma nella zona NUOVA, che indica la zona dove le applicazioni accedono perché vogliono accedere ai dati. E POI INCREMENTIAMO IL CONTATORE.

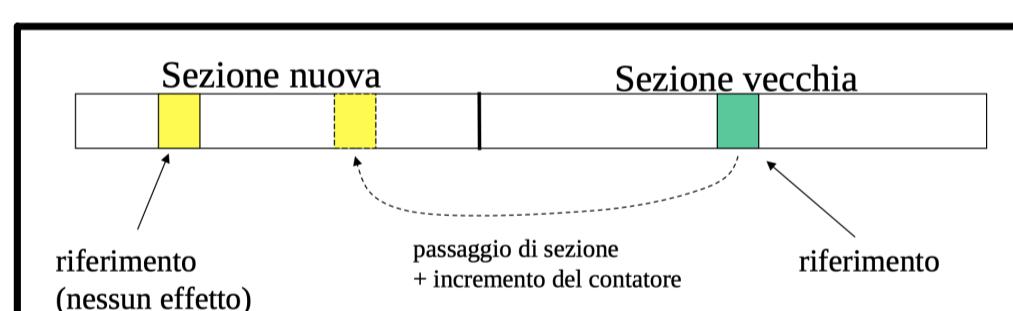
Quand'è che viene incrementato il contatore di un elemento che è già in buffer-cache?

Se l'elemento è nella zona VECCHIA.

Era stato caricato precedentemente nella zona NUOVA, poi questa era satura ed è stato spostato nella zona vecchia.

Carichiamo sicuramente un elemento nella zona nuova, ma poi lo spostiamo in zona vecchia, quando dobbiamo caricare altri blocchi nella zona nuova, in sostanza facciamo spazio.

Però se il blocco è in zona vecchia e ci serve di nuovo? Lo riportiamo tranquillamente in zona nuova, ma qualcosa dovrà uscire da quest'ultima.



Quando riportiamo un elemento nella zona nuova, ovviamente, re-incrementiamo il contatore.

Il contatore d'uso di un blocco associato ai dati di un certo FILE viene incrementato ogni volta che noi andiamo a riportare il blocco / o a portare il blocco per la prima volta, nella sezione nuova.

Quindi il contatore sta contando quante volte il blocco è stato portato nella sezione nuova.

Chiaramente quando dobbiamo fare spazio ad un nuovo elemento da inserire nella sezione nuova, noi portiamo fuori da questa sezione il blocco con il contatore minore, stessa cosa se dalla sezione vecchia dobbiamo liberare qualcosa perché stiamo portando qualcosa dalla sezione nuova, riportiamo sul dispositivo di memoria di massa il blocco con contatore minore.

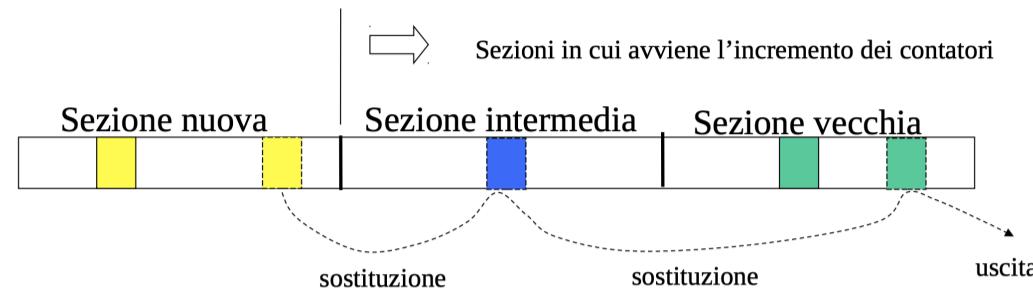
Un blocco con contatore alto è un blocco che ancora serve alle applicazioni. Abbiamo risolto il problema.

- ogni volta che un blocco è riferito, il suo contatore di riferimenti è incrementato ed il blocco è portato nella **sezione nuova**
- per i blocchi nella sezione nuova il contatore di riferimenti non viene incrementato
- per la sostituzione dalla sezione nuova si sceglie il blocco con il numero di riferimenti minore
- la stessa politica è usata per la sostituzione nella **sezione vecchia**

Le sezioni possono anche essere più di due:

Buffer cache a tre sezioni

- esiste una sezione intermedia
- i blocchi della sezione intermedia vengono passati nella sezione vecchia per effetto della politica di sostituzione
- un blocco della sezione nuova difficilmente verrà escluso dal buffer cache in caso sia riferito in breve tempo dall'uscita dalla sezione nuova



Ci spostiamo nella sezione intermedia quando abbiamo il contatore più basso rispetto a tutti i blocchi della sezione nuova, e dalla sezione intermedia usciamo e andiamo verso la sezione vecchia quando abbiamo il contatore più basso di tutti gli elementi della sezione intermedia. Usciamo dal BUFFER CACHE quando abbiamo il contatore più basso della sezione vecchia. Un contatore viene ad essere incrementato quando un blocco sia della sezione intermedia che vecchia, viene ad essere in sezione nuova, quindi viene acceduto nuovamente.

- esiste una sezione intermedia
- i blocchi della sezione intermedia vengono passati nella sezione vecchia per effetto della politica di sostituzione
- un blocco della sezione nuova difficilmente verrà escluso dal buffer cache in caso sia riferito in breve tempo dall'uscita dalla sezione nuova