Semantica della consistenza sul file system

venerdì 21 aprile 2023 17:33

Semantica della consistenza sul file system

Quando noi lavoriamo all'interno di un file system abbiamo sicuramente il dispositivo fisico dove i dati sono presenti, quindi il disco rigido dove ci sono i blocchi di dati dei file, però in RAM abbiamo il buffer cache e un aggiornamento di un file f fa accadere questo: Portiamo il blocco che contiene i dati dal dispositivo di memoria di massa in buffer cache, e poi LO aggiorniamo in base ai dati che ci vengono passati dalle applicazione tramite le system call. Ci manteniamo i dati aggiornati all'interno del buffer cache fino a che non è passato un certo intervallo di tempo o sono stati certi aggiornamenti, oppure non c'è più spazio in buffer cache. Poi lo riportiamo nel dispositivo di memoria di massa aggiornando la versione di informazioni che può sopravvivere ad un power off. Però supponiamo che noi abbiamo mandato il blocco in buffer cache all'istante T0, e dopo lo rimandiamo in memoria di massa a T1, che succede in questo frangente? Può succedere qualsiasi bug a livello del sistema operativo. I dati che sono in ram, rimangono in ram e non sono correttamente portati sul dispositivo di memoria di massa. Quindi ciò che abbiamo scritto all'interno di un file in realtà è perso.
Quindi sul dispositivo di memoria di massa D, l'informazione aggiornata non c'è mai andata.
Quindi ci troviamo un file che è inconsistente.

Semantica UNIX

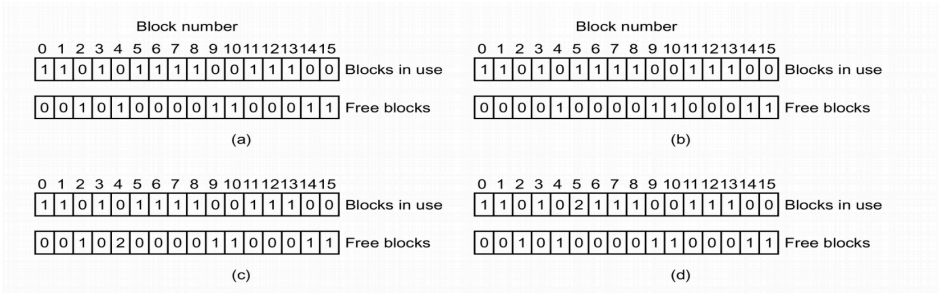
- ogni scrittura di dati sul file system è direttamente visibile ad ogni lettura che sia eseguita successivamente (riletture da buffer cache)
- supportata anche da Windows NT/2000/....
- solo approssimata da NFS (Network File System)

Semantica della sessione

- ogni scrittura di dati sul file system è visibile solo dopo che il file su cui si è scritto è stato chiuso (buffer cache di processo)
- supportata dal sistema operativo AIX (file system ANDREW)

- crash di sistema possono portare il file system in uno stato inconsistente a causa di
  1. operazioni di I/O ancora pendenti nel buffer cache
  2. aggiornamenti della struttura del file system ancora non permanenti
- possibilità ricostruire il file system con apposite utility
  - *fsck* in UNIX
  - *scandisk* in Windows
- vengono analizzate le strutture del file system (MFT, I-nodes) e si ricava per ogni file system:
  - *Blocchi in uso*: a quanti e quali file appartengono (si spera uno eccetto che in scenari di deduplicazione)
  - *Blocchi liberi*: quante volte compaiono nella lista libera (si spera una o nessuna)

Ricostruzione dei file system



a) Situazione consistente

b) *Il blocco 2 non è in nessun file né nella lista libera*: aggiungilo alla lista libera

c) *Il blocco 4 compare due volte nella lista libera*: togli un'occorrenza

d) *Il blocco 5 compare in due file*: duplica il blocco e sostituiscilo in uno dei file (rimedio parziale!!!)

Sincronizzazione del file system UNIX

```
NAME
       fsync,  fdatasync - synchronize a file's complete in-core state with
       that on disk
```

SYNOPSIS

```
#include <unistd.h>

int fsync(int fd);

int fdatasync(int fd);
```

DESCRIPTION

fsync copies all in-core parts of a file to disk, and waits until the device reports that all parts are on stable storage. It also updates metadata stat information. It does not necessarily ensure that the entry in the directory containing the file has also reached disk. For that an explicit **fsync** on the file descriptor of the directory is also needed.

fdatasync does the same as **fsync** but only flushes user data, not the meta data like the mtime or atime.

LE POSSO SPECIFICARE ANCHE DENTRO UN PROGRAMMA APPLICATIVO: è una richiesta al kernel di sincronizzare i dati che sono su un dispositivo di memoria di massa rispetto agli aggiornamenti di questi dati che sono presenti all'interno della ram (buffer cache o cache degli I-NODE.)

Prendere i dati dal buffer cache e di flasharli in memoria di massa. Se andiamo in shutdown sicuramente quando riaccendiamo questi dati rimangono.