File System Journal

Oleh:
Rahmad Wahyudi
Elektronika dan Instrumentasi
Universitas Gadjah Mada
Yogyakarta

Abstrak

Pencatatan (journaling) merupakan salah satu keistimewaan dari file systems yang modern. Suatu file system journal akan men-cache data yang akan dituliskan ke file system untuk memastikan agar data tersebut tidak hilang ketika terjadi power loss atau system malfunction. Menganalisa journal data dapat membantu dalam mengidentifikasi file-file mana yang baru saja di-overwrite. Pada kondisi-kondisi tertentu, dengan menganalisa journal file systems kita dapat mengetahui atau melihat file-file yang sudah dihapus ataupun previous version dari suatu file tanpa harus melakukan review pada hex dump dari sebuah drive. Pembahasan kali ini adalah mengenai journal pada file system ReiserFS dan ext3.

I. Pendahuluan

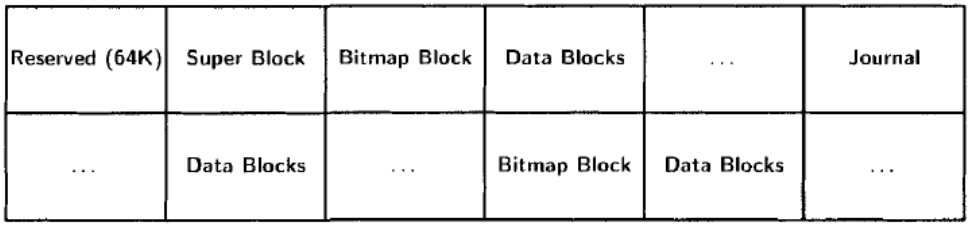
Pencatatan (journaling) adalah suatu kemajuan teknologi dari file systems. Fitur ini bekerja pada semua file systems yang modern, termasuk NTFS (Windows-NT/2000/XP), HFSJ (Mac OS X), ext3 (Linux), dan ReiserFS (Linux).

Suatu file system journal bekerja dengan cara men-cache sebagian atau seluruh data yang ditulis pada suatu bagian dari disk sebelum ditulis ke file system. Ketika suatu saat terjadi power loss, malfunction, atau sesuatu yang tidak normal, journal tersebut dapat diulangi kembali untuk menyelesaikan penulisan yang belum selesai sehingga dapat menghindari file system corruption karena operasi penulisan yang belum selesai. Ini berarti bahwa previous file akan disimpan dengan periode terbatas di luar normal file system. Karena itu, walaupun suatu file tertimpa (overwritten) atau terhapus, dimungkinkan untuk mengembalikan lagi konten dari file tersebut dengan menganalisa journal dari suatu file system.

II. Dasar Teori

1. Struktur ReiserFS

ReiserFS mempunyai suatu struktur blok dengan ukuran blok-blok yang tetap (biasanya 4.096 byte). Struktur blok dari ReiserFS adalah seperti gambar berikut :



Blok pertama pada struktur ReiserFS adalah superblock. Tidak seperti file system yang lainnya (misalnya ext2), ReiserFS hanya mempunyai sebuah superblocks. Struktur dari superblock tersebut ditunjukkan pada *tabel 1* di bawah.

ReiserFS bitmap blocks adalah blok-blok khusus untuk mengidentifikasi blok yang sudah terpakai dan yang belum terpakai. Setiap bit pada bitmap block bertindak sebagai "used bit" untuk suatu single block pada file system.

ReiserFS mempunyai tiga jenis data blocks yaitu, unformatted data blocks, internal data blocks, dan leaf blocks.

tabel 1

Name	Bytes	Description
Block Count	4	Number of blocks in file system
Free Blocks	4	Number of unallocated blocks
Root Block	4	Location of the root block
Journal Information	28	Various aspects of the journal
Block Size	2	File system block size
Object ID Max. Size	2	Maximum size of the OIDs
Object ID Current Size	2	Current size of the OIDs
State	2	Whether the partition is clean
Magic String	12	`ReIsEr2Fs`
Hash Function	4	File name hash function
Tree Height	2	Height of file system B-tree
Bitmap Number	2	Number of bitmap blocks
Version	2	Version of the superblock
Reserved	2	Reserved
Inode Generation	4	Rebalancing count

2. Struktur ext3

File system ext3 merupakan pengembangan dari file system ext2 yang sebelumnya, dengan menambahkan fitur journaling. Ext3 menjadi file system yang terpopuler di Linux, dan biasanya menjadi pilihan default untuk suatu installasi baru.

Seperti ReiserFS, ext3 juga menggunakan struktur blok. Ukuran blok di seluruh file system sama, bergantung pada ukuran file system tersebut, meskipun demikian, harga default-nya adalah antara 1.024 atau 4.096 byte. 1.024 byte pertama dari ext3 file system selalu terpakai. Jika di dalam file system ada boot kernel, maka byte-byte ini akan berisi boot information.

Superblock pada ext3 file system menyimpan informasi-informasi umum, misalnya nama, ukuran, dan data kapan terakhir kali dipakai. Ext3 file system hanya mempunyai satu primary superblock, walaupun dapat dibuat backup copy-nya juga dapat disimpan di sepanjang file system. Struktur dari ext3 superblock dapat dilihat pada *tabel 2* di bawah.

File system ext3 menaruh file metadata dalam suatu struktur data yang disebut inodes. Inodes disimpan pada blok-blok khusus yang dinamakan inode tables. Panjang standar dari suatu inode adalah 256 byte. Setiap struktur inode dapat menampung hingga dua belas direct pointer, yang berhubungan dengan alamat-alamat dari blok file system dimana dua belas blok yang pertama dari suatu file diletakkan. Jika file terlalu besar untuk diletakkan pada dua belas blok pertama (biasanya 12 KB atau 48 KB), maka pointer akan dijadikan ke single, double, dan triple indirect pointer blocks. Suatu single indirect pointer adalah alamat dari blok file system yang terdiri dari semua direct pointer. Selanjutnya, double dan triple indirect pointer menunjuk kepada blok file system yang mengandung single dan double idirect pointer. Struktur detail dari ext3 inode dapat dilihat pada *tabel 3.*

tabel 2

Name	Bytes	Description
Inode Count	4	Total number of inodes in file system
Block Count	4	Total number of blocks in file system
Blocks Reserved	4	Reserved block count to prevent overfill
Free Block Count	4	Number of unallocated blocks
Free Inode Count	4	Number of unallocated inodes
Group 0	4	Block where first block group starts
Block Size	4	Left shifts of 1024 to obtain block size
Fragment Size	4	Left shifts of 1024 to obtain fragment size
Blocks per Block Grp.	4	Blocks in a typical block group
Fragments per Block Grp.	4	Fragment count in a typical block group
Inodes per Block Grp.	4	Inodes in a typical block group
Last Mount Time	4	Seconds from epoch to last mount time
Last Written Time	4	Seconds from epoch to last write time
Mount Information	4	Total and max. mounts of file system
Signature	2	0xEF53
File System State	2	Clean, error, recovering orphan inodes
Error Handling Method	2	Continue, remount as read only, or panic
Minor Version	2	Original or dynamic
Consistency Check	8	Last performed, interval
Creator OS	4	Linux, FreeBSD, etc.
Major Version	4	Original or dynamic
Reserved Block UID/GID	4	UID/GID that can use reserved blocks
First Inode	4	First non-reserved inode in file system
Inode Size	2	Size of inode in bytes
Block Grp. Loc. of Copy	2	If backup copy, group of copy
Feature Flags	12	Features of the file system
File System ID	16	UUID of file system
Volume Name	16	OS's name for the volume
Other Misc. Information	72	Misc.
Journal Information	24	UUID, metadata inode, device
Orphan Inodes	4	Head of orphan inode list
Unused	788	Unused bytes

tabel 3

Name	Bytes	Description
File Mode	2	Permission flags and file type
User ID	2	Lower 16 bits of user ID
File Size	4	Lower 32 bits of size in bytes
ACMD Times	16	Most recent access, creation, mod., del. times
Group ID	2	Lower 16 bits of group ID
Link Count	2	Number of existing links to the file
Sector Count	4	Sector occupied by the file
Flags	4	Assorted flags
Unused	4	Unused bytes
Direct Pointers	48	12 direct pointers
Single Indirect Pointer	4	1 single indirect pointer
Double Indirect Pointer	4	1 double indirect pointer
Triple Indirect Pointer	4	1 triple indirect pointer
Misc. Information	8	NFS gen. number, extended attribute block
File Size	4	Upper 32 bits of size in bytes
Fragment Information	9	Address, count and size
Unused	2	Unused bytes
User ID	2	Upper 16 bits of user ID
Group ID	2	Upper 16 bits of group ID
Unused	2	Unused bytes

Blok-blok pada ext3 disusun menjadi block groups. Block groups diuraikan dalam suatu blok atau sekelompok blok bernama “the group descriptor table”, yang selalu mengikuti superblock atau copy dari superblock.

Setiap block group berisi dua bitmap block, satu untuk blok dan yang lainnya untuk inode. Setiap bit dalam blok bitmap menunjukkan status dari salah satu blok atau inode dalam group tersebut apakah allocated atau unallocated. Blok dan inode berhubungan pada older version dari suatu content file atau suatu file yang terhapus. Blok yang belum pernah dialokasikan diwakili dengan “0”, sedangkan yang berisi content dari suatu file atau metadata dan yang dipakai oleh file system diwakili dengan “1”.

III. Pembahasan

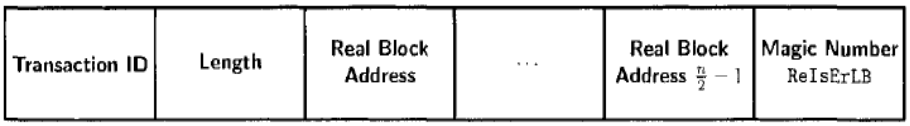
Suatu journal, pada umumnya berisi raw blok yang akan dituliskan ke hard disk. Blok-blok ini dapat berupa unformatted user data atau mungkin blok-blok dalam struktur internal dari suatu file system tree. Ketika suatu file dimodifikasi pada drive, blok-blok berisikan raw data, baik secara terpisah atau di dalam metadata itu sendiri, ditulis ulang bersama dengan metadata. Dengan menggunakan peta blok (block map) pada permulaan journal dapat membantu menentukan blok-blok mana yang berhubungan dengan suatu file tertentu.

Adalah suatu hal yang penting untuk mengingat bahwa journal berisi lebih dari sekedar file-file yang terhapus. Previous versions dari suatu file juga dapat ditemukan bersama modifikasi yang terakhir. Sementara MAC times menayampaikan hanya jika file dimodifikasi, dibuat, atau diakses, journal melacak bagian mana dari file tersebut yang terakhir dimodifikasi.

1. **Journal pada ReiserFS**

   ReiserFS dalam Linux mempunyai ukuran journal tertentu yang terdiri dari 8.192 byte blok, ditambah 4.096 byte blok header, dengan ukuran total sekitar 32 MB. Blok header menggunakan dua belas byte pertama untuk selalu memantau blok mana yang merupakan potongan terakhir dan dimana untuk memulai potongan selanjutnya, juga sebagai mount information.

   Remainder dari suatu journal terdiri dari sejumlah catatan (transaction) dalam urutan melingkar. Setiap catatan dimulai dengan description block, yang mengidentifikasi catatan dan menyediakan separuh dari peta yang menunjukkan dimana untuk menaruh blok data. Catatan tersebut diakhiri dengan blok akhir yang berisi bagian yang tersisa dari peta blok dan hubungan dari catatan tersebut seperti pada gambar berikut :



   Journal pada ReiserFS mempunyai ukuran standar 32 MB, yang cukup untuk jumlah data yang besar, termasuk dokumen, gambar-gambar, dan materi lainnya. Menghapus file pada ReiserFS pada umumnya tidak akan membersihkan journal, sekalipun dengan utility penghapusan.

   Catatan (transaction) dalam ReiserFS mengandung seluruh blok data yang akan ditulis ke hard drive, berfungsi sebagai cache pada komputer. Selanjutnya, catatan dalam cache tidak akan dihapus sampai terjadi overload. Jadi catatan tersebut akan ada cukup lama dan berisi copy dari recent data.

2. **Journal pada ext3**

   Ukuran journal pada file system ext3 bergantung pada ukuran file system seperti pada tabel di bawah ini. Secara default, hanya metadata yang disimpan di journal.

File System Size	Block Size	Journal Blocks	Journal Size
< 2 MB	1,024 B	0	0 MB
2 MB	1,024 B	1,024	1 MB
32 MB	1,024 B	4,096	4 MB
256 MB	1,024 B	8,192	8 MB
512 MB	1,024 B	16,384	16 MB
513 MB	4,096 B	4,096	16 MB
1 GB	4,096 B	8,192	32 MB
2 GB	4,096 B	16,384	64 MB

   Blok pertama pada journal selalu berisi journal superblock khusus yang menyimpan informasi spesifik dari journal tersebut (*tabel 4*). Isi dari journal tersebut disimpan dengan urutan melingkar dan setiap isian mempunyai satu blok masukan dan paling sedikit satu descriptor block (*tabel 5*). Descriptor block menyediakan urutan-urutan

dari catatan (transaction) dan blok-blok file system yang disimpan. Jika catatan memuat lebih banyak blok dari yang dapat dimuat dalam satu descriptor block, maka descriptor block yang lain akan dibuat untuk menampung sisa blok.

Bila file system pada data journaling mode, new versions dari suatu data dituliskan pada journal sebelum ditulis ke disk. Seperti pada ReiserFS journal, jornal pada ext3 juga dapat menyediakan informasi-informasi tentang content dari file yang tua, terhapus, atau termodifikasi.

Pada file system ext3 dengan pengaturan journal default, hanya perubahan pada metadata yang ditulis di journal. Semua blok-blok non-journal dicatat hanya bila ada modifikasi. Oleh sebab itu, jika sebuah file di-edit, semua metadata dicatat bersama content file yang baru. Walaupun metadata ini dapat sangat berguna untuk data recovery, hanya blok-blok data yang tersimpan di journal saja yang dapat digunakan untuk keperluan ini. Untuk memungkinkan journaling semua data, diperlukan untuk memuat file system dengan option data=journal (di Linux).

tabel 4

Name	Bytes	Description
Header	12	Signature (0xC03B3998), block type
Block Size	4	Size of journal block in bytes
Block Count	4	Number of blocks in journal
Start Block	4	Block where journal starts
First Transaction Sequence	4	Sequence number of the first transaction
First Transaction Block	4	Journal block of first transaction
Error Number	4	Information on errors
Features	12	Features of the journal
Journal UUID	4	Universally unique identifier of the journal
File System Count	4	Number of file systems using journal
Superblock Copy	4	Location of superblock copy
Journal Blocks per Trans.	4	Max. journal blocks per transaction
FS Blocks per Trans.	4	Max. FS blocks per transaction
Unused	176	Unused bytes
FS IDs	768	IDs of file systems using the journal

tabel 5

Name	Bytes	Description
Header	12	Sig. (0xC03B3998), seq. num., block type
File System Block	4	File system block where content will be written
Entry Flags	4	Same UUID, last entry in descriptor block, etc.
UUID	16	Only exists if the SAME_UUID flag is not set

IV. Kesimpulan

Karena suatu journal dari file system men-cache data mengenai file yang ditulis, maka menganalisa journal dapat menyediakan informasi yang berharga tentang versi-versi yang lau dari sebuah file. Pada kondisi yang benar, suatu journal dapat menjabarkan informasi-informasi tentang file yang terhapus dan versi lalu (previous version) dari suatu file tanpa harus melakukan review hex dump dari suatu drive. Journal juga dapat memberikan bukti mengenai file-file yang tertumpuk (overwriteten) atau dihapus dengan deletion utility.

V. Referensi

1. F.Buchholz. *The structure of the Reiser file system*. (homes.cerias.purdue.edu/~florian/reiser/reiserfs.php)
2. R. Card, T. Ts'o and S. Tweedie.1994. *Design and Implementation of the Second Extended File System*. *Proceedings of the First Dutch International Symposium on Linux* (web.mit.edu/tytso/www/linux/ext2intro.html)
3. M. Rosenblum and J. Ousterhout.1992.*The Design and Implementation of a log-structured File System.ACM Transactions on Computer Systems*.vol. 10(1), pp. 26-52
4. Christopher Swenson, Raquel Philhps and Sujeet Shenoi. *File System Journal Forensics.*