Design of Recoverable / Fault Tolerant File System

Table of Content

Existing File System: EXT4

The chosen file system is EXT4 (Extended File System) which is used by default in most Linux distributions.
The salient features of EXT4 filesystem is as follows:

- There is a boot sector in the first sector of the hard drive which includes a very small boot record and a partition table. Then there is some reserved space after the boot sector, which spans the space between the boot record and the first partition on the hard drive that is usually on the next cylinder boundary. GRUB —uses this space for part of its boot code.
- The space in each partition is divided into cylinder groups that allow for more granular management of the data space. Figure 1, below, shows the basic structure of a cylinder group. A block is a data allocation unit in a cylinder, which is usually 4K in size.

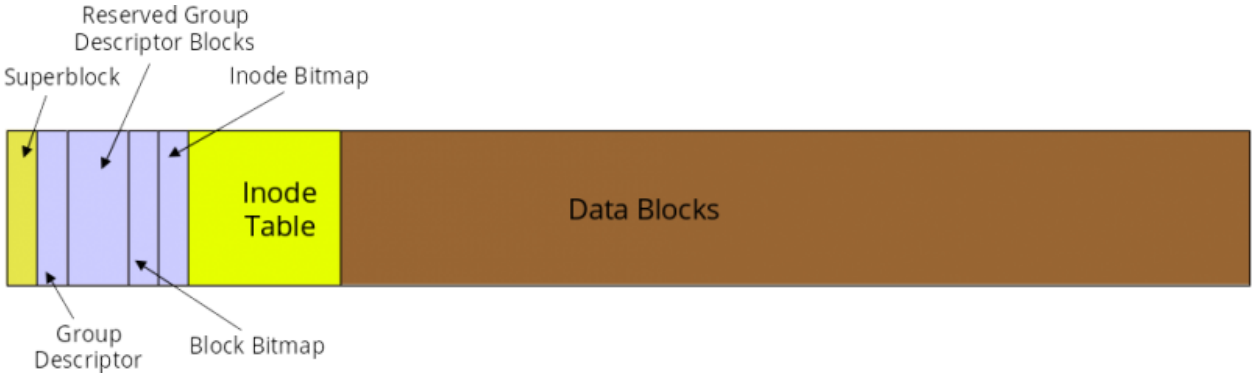

Fig. 1: Structure of Cylinder Group

- The first block in the cylinder group is a superblock , which contains the metadata that defines the other file system structures and locates them on the physical disk. Extended File System creates backup copies of some of the super blocks located at different partitions of the disk.
- Each cylinder group has its own inode bitmap that is used to determine which inodes are used and which are free within that group. The inodes have their own space in each group. Each inode contains information about one file, including the locations of the data blocks belonging to the file. The block bitmap keeps track of the used and free data blocks within the filesystem.

Bad Block

A bad block is that area of the hard disk that becomes unreliable for storing and retrieving data. The possible causes are physical damage or data corruption. Bad blocks are also referred to as bad sectors.

There are two types of bad blocks :

- A physical, or hard, bad block is the one which occurs due to the damage of the storage medium or when the recording surface is defective.
- A soft, or logical, bad block occurs when the operating system (OS) is unable to read data from a sector. It happens when the cyclic redundancy check , or error correction code , for a particular storage block does not match the data read by the disk.

Corrupted data block is undesirable for any file system because vital data gets lost. The filesystems are designed to ensure data recovery at the time of data failure.

Proposed Design for minimizing information loss

Structure of a block:

Every block is made up of 2 parts. One is the data and second is metadata of the file. The metadata consists of 2 fields:

1) **File id:** A unique number assigned to every file at the time of its creation.
2) **Block number:** A file is stored in a number of blocks. Each block is assigned a number in sequence of its occurrence in the file, for example, B0 denotes the first block of the file.

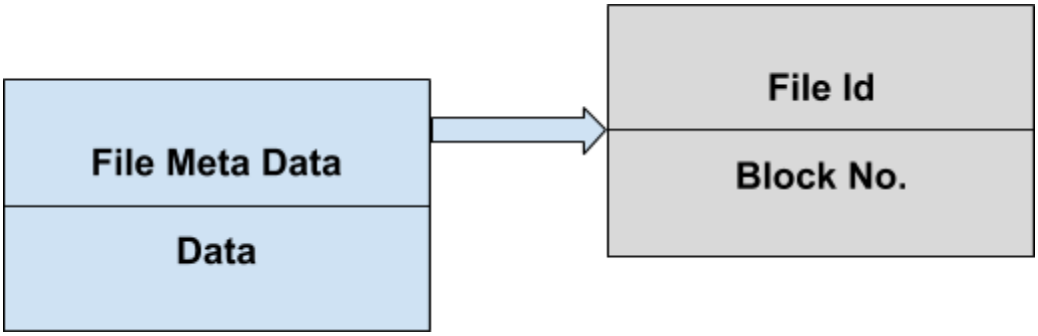

Fig. 2: Structure of a block

Recovery of information:

The following cases arise:

I. The data block becomes inaccessible:

In this case, the information present in that particular block is lost while the information in other blocks remain intact. The inode block can still be used to access other blocks.

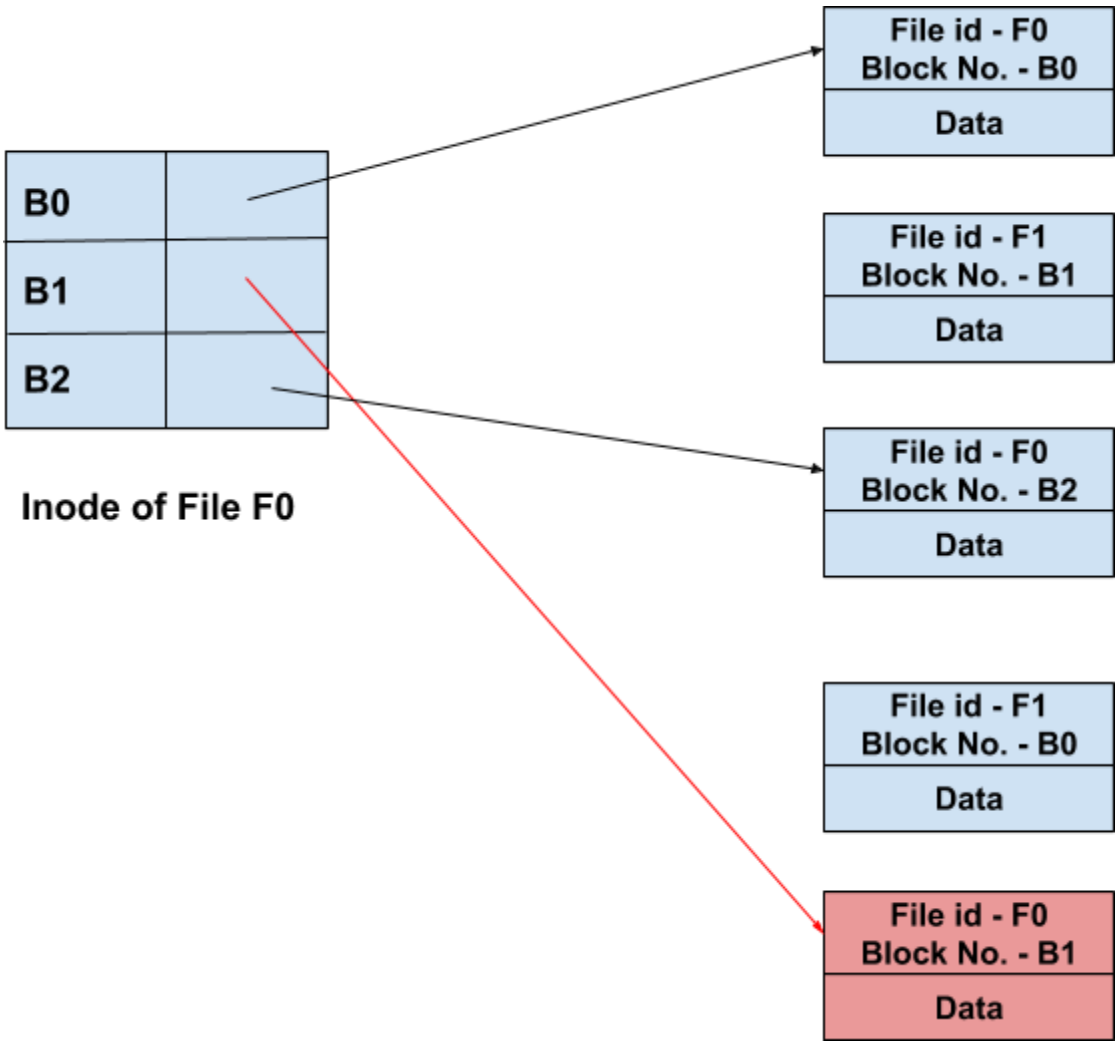

Fig. 3: Data block inaccessible

In the diagram above, the last block i.e. File ID F0 and Block number B1 has become inaccessible. The data present in this block will be lost but we can use the inode of F0 to access the information in all the remaining blocks. This guarantees minimal loss of information.

II. Inode block becomes inaccessible:

Since we have lost the inode, we need to recreate it. To recreate the inode, we have designed a traceback algorithm.

Traceback algorithm:

We begin scanning the disk from the start and we read the metadata of every block. If the File Id matches the file id of the lost inode, we create an entry in our newly recreated inode. Repeat this process until we exhaust scanning the whole disk. At the end of this process, we would have traced out out the lost inode.

The diagrams below explain the process:

1) Inode of File F0 becomes inaccessible. We start the scan from the first block and trace back the inode.

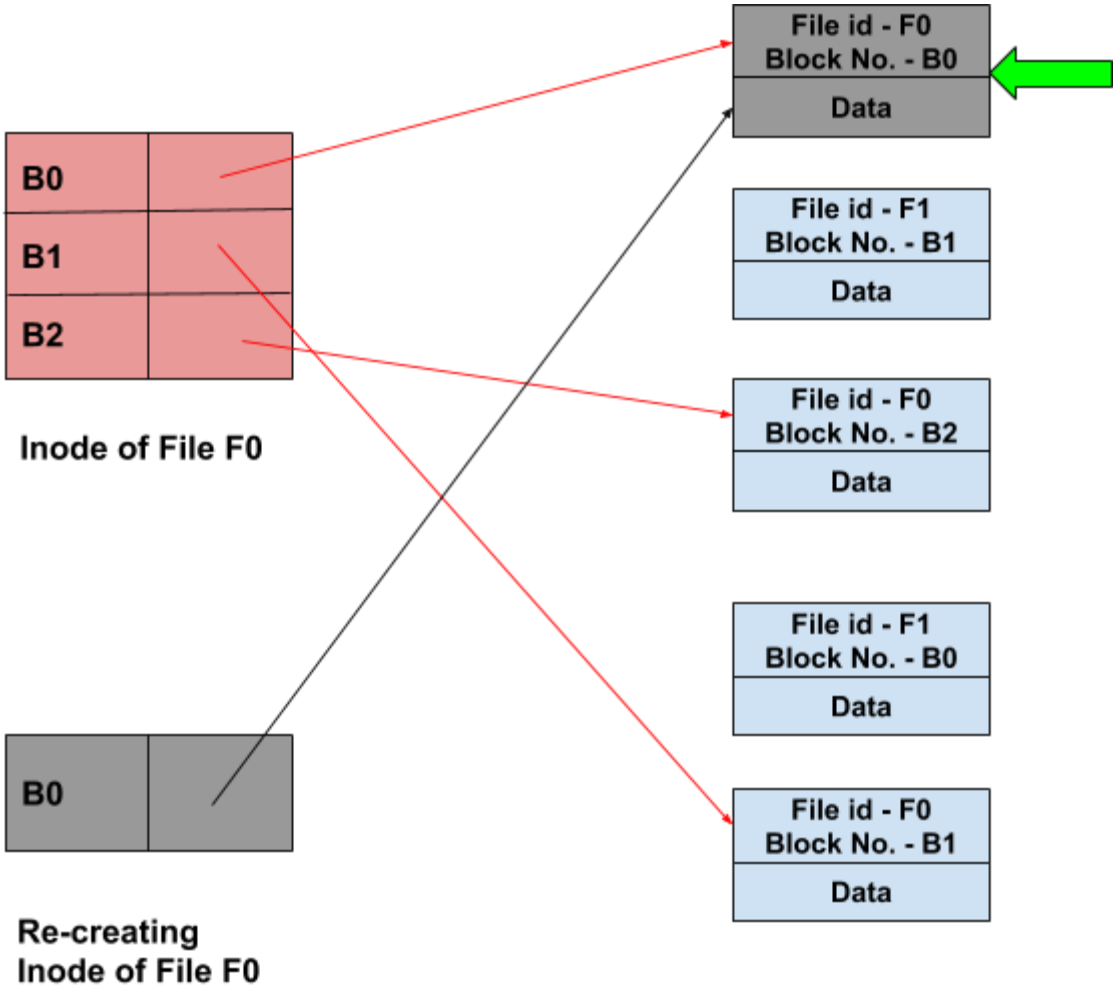

Fig. 4: Inode becomes inaccessible

2) Scan the next block and if file id doesn't match we move ahead to the next block until the end of disk is reached.

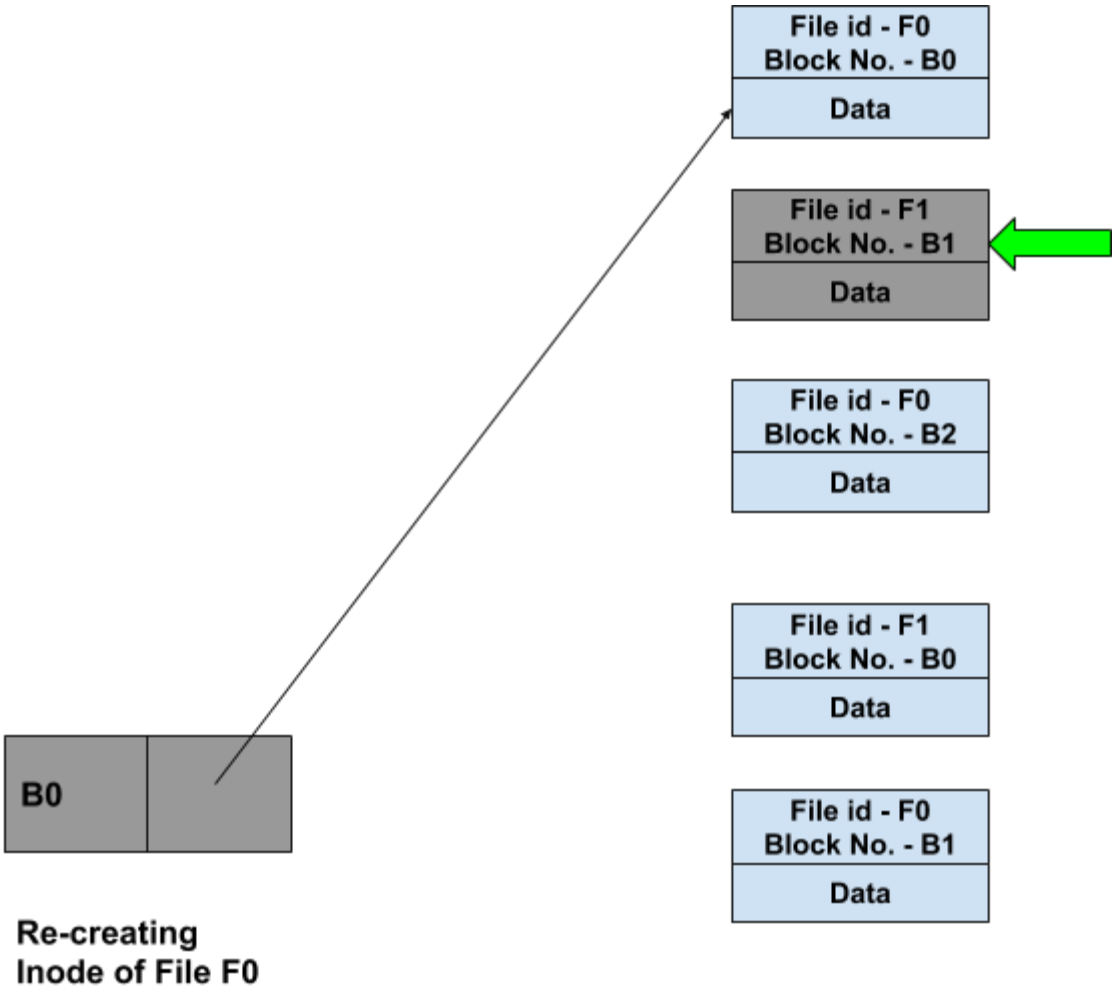

Fig. 5: Trace back

3) Keep scanning until the end of the disk is reached and when file id matches create a new entry in inode.

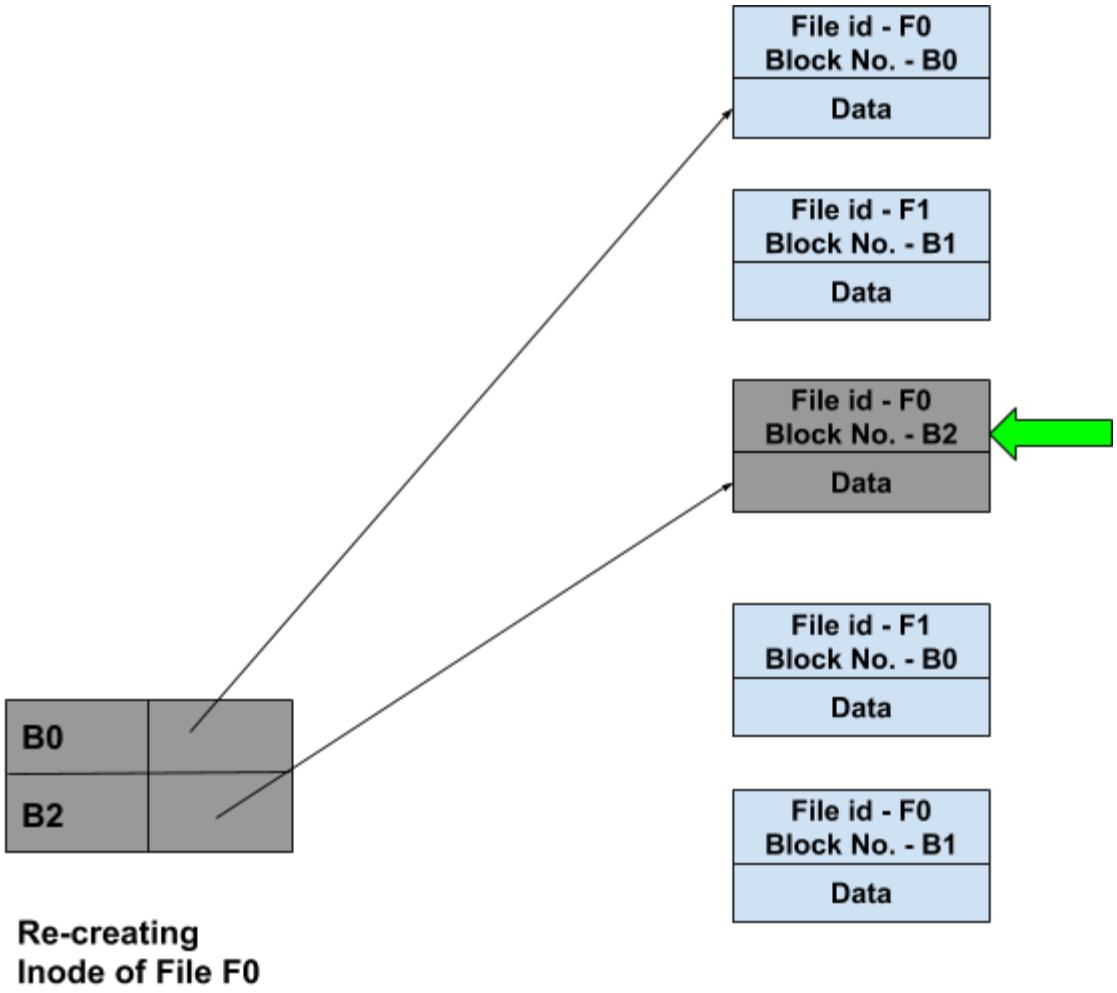

Fig. 6: Inode for F0 recreated

Pros:

1. Disk space can be fully utilised as we are not storing any copies.
2. Easy and intuitive algorithm.

Cons:

1. Time taken to reconstruct the inode can be very high since we are scanning the disk from start till the end.

Completely Fault tolerant file system

- In a completely fault tolerant system, any data can be recovered regardless of hardware failure, data corruption, software errors or operator mistakes.
- When a block is corrupted or inaccessible, then file system should be able to recover that block's data from somewhere.
- Since now a days, data storage capacity is not a big issue and storage capacity to cost ratio is increasing, Studies say that in general not even half of the disk is used by the users to store the files. So an inexpensive approach to make a fault tolerant file system is to somehow efficiently use this unused space of the disks.
- It could also be extended to be able to prevent the files from catastrophic failures.

Design:

- Storage disk is divided into 2 equal parts : Primary access part and Backup Space (to keep replica of primary access part).
- A log file is maintained to keep track of blocks in primary part of disk on which changes have been made since the last backup.
- Further, 1 bit is reserved in each block to denote whether some changes have been made in the data of that block since the last update. This bit is called **dirty bit**.

Approach to read data blocks:

- The system will first try to read from the primary space on the disk,
- In case of any inconsistency or corrupted block, system will read/load the corresponding content from the backup space.

Approach to write data blocks:

- Write data in the primary part of disk.
- Update the log file to keep track of all the write updates being done on the original data blocks.
- After periodic intervals, the backup space are updated based on the log file, to keep the blocks consistent.
- In case log file is corrupted, the dirty bits stored at each block could be traced for performing backup and maintain consistency.

In case of catastrophic failures, keep a backup of data on a separate storage like NAS which are RAID compliant. This added redundancy can help recover all the data without any information loss.

Pros:

1. Data lost due to any damage or corruption can be recovered if the block gets corrupted/damaged.

Cons

1. Memory space available to users gets reduced.
2. Organizing the backup space and recovering the data item can be a complicated process.
3. Overhead involved while logging.