OSFanatics - CS 270

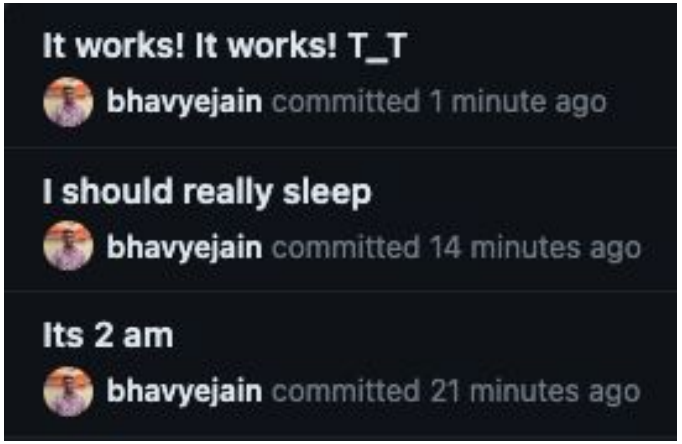
AltFS

Bhavye Jain
Swathi Bhat

File-tastical follies

What we lost and learnt through our odyssey

- Code only as good as tests
- Logs for life!
- #SleepDeprivedButOurFileSystemThrived (hopefully :))

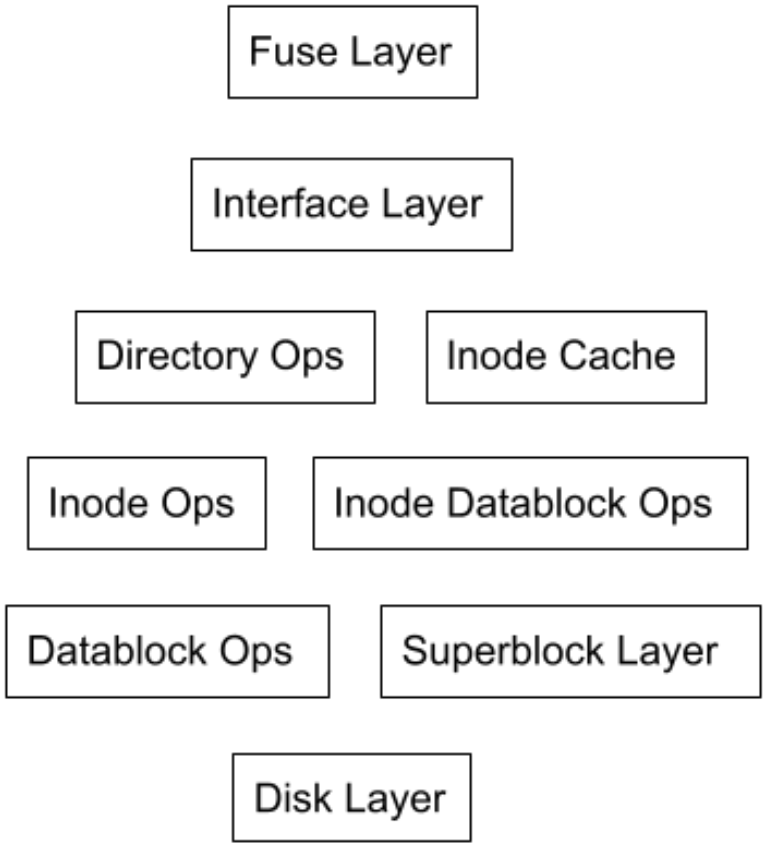


Because 1 filesystem just wasn't enough…

We built & mounted our FS inside our FS and it works!!

```
[root@ip-172-31-19-208 mnt]# pwd
/home/cloud-user/AltFileSystem/disk/AltFileSystem/mnt
[root@ip-172-31-19-208 mnt]# mkdir dir1
[root@ip-172-31-19-208 mnt]# touch dir1/file.txt
[root@ip-172-31-19-208 mnt]# echo "Hello world!" > dir1/file.txt
[root@ip-172-31-19-208 mnt]# cat dir1/file.txt
Hello world!
[root@ip-172-31-19-208 mnt]#
```

Code structure for AltFS



Filesystem config parameters

- Block size : 4096 B
- Inode structure

```
/*
Follows a structure similar to ext4.
(https://www.kernel.org/doc/html/latest/filesystems/ext4/inodes.html?highlight=inode)
*/
struct inode
{
    mode_t i_mode; // permission mode
    id_t i_uid; // lower 16-bits of owner id
    id_t i_gid; // lower 16-bits of group id
    time_t i_status_change_time; // status change time
    time_t i_atime; // last access time
    time_t i_ctime; // last inode change time
    time_t i_mtime; // last data modification time
    nlink_t i_links_count; // hard link count
    ssize_t i_file_size; // file size
    ssize_t i_blocks_num; // num of blocks the file has
    bool i_allocated; // flag to indicate if inode is allocated
    // TODO: Kept number of direct blocks as 12 in sync with ext4
    ssize_t i_direct_blocks[NUM_OF_DIRECT_BLOCKS];
    ssize_t i_single_indirect; // stores block num for single indirect block
    ssize_t i_double_indirect; // stores block num for double indirect block
    ssize_t i_triple_indirect; // stores block num for triple indirect block
    nlink_t i_child_num; // stores the current number of entries for a directory (minimum 2) or 0 for others
    char padding;
};
```

- Inode blocks : 1.5% of total blocks
- Inode cache capacity : 100000 records

Proud to call it ours

Proud to call it ours

Makefs

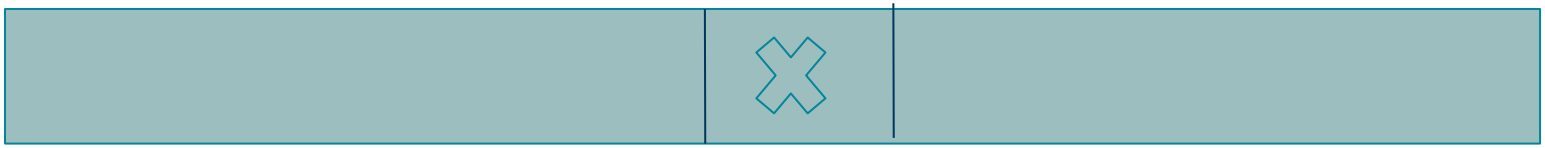
- Iteration 1: Erasing disk contents and formatting as part of initialization.
  - Erase required to prevent file corruption.
- Iteration 2: A separate mkaltfs utility and ability to load from disk on startup.
  - Persist data on unmount and remount.
- Iteration 3: Remove requirement for erasing disk while formatting!
  - Mkaltfs gets options!

Proud to call it ours

Directory records

- No holes in a data block
- Record length = 0 => no directory entries from that point onwards

- Add new entry - search for first record where record length = 0
- Remove entry - Copy rest of the block starting from the next entry into buffer, delete the current entry and move the rest of the blocks left. This leaves no holes

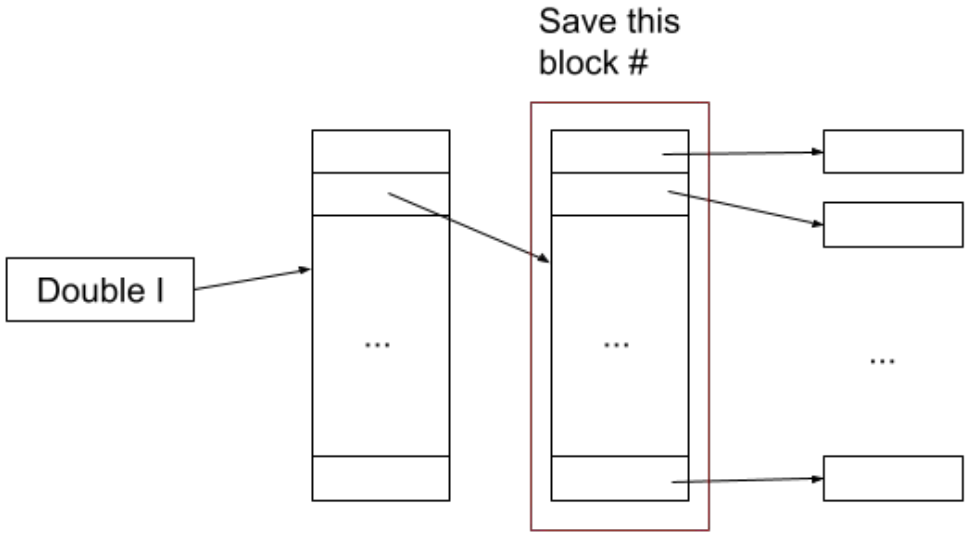


- Get file position in directory - Check for each block until record len = 0 and if file name matches given name

Proud to call it ours

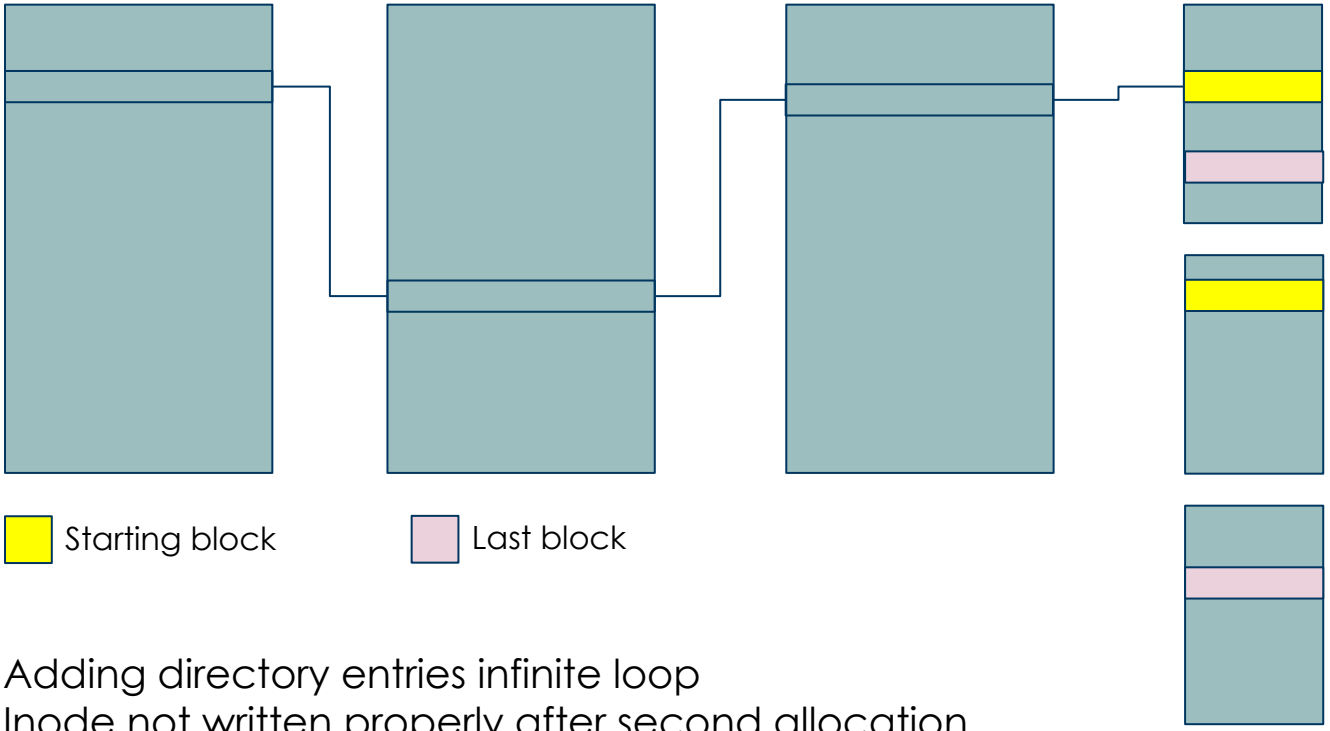
Inode data block traversal

- Fetching double and triple indirect blocks is expensive!
- Make it stateful!



How our tests helped better our code

- Remove data blocks from inode starting from a given logical number



- Adding directory entries infinite loop
- Inode not written properly after second allocation
- Pointer freeing in cache flush
- operating vim and gcc - writing more bytes than requested

Limitations

- Data locality - too many seeks!
- Finding next free inode is expensive.
- Permissions are partially supported.
- Symlinks not supported.

UC SANTA BARBARA