Week 11.b
CS3650
03/20 2024
https://naizhengtan.github.io/24spring/

1. fs updates
2. Crash recovery
   --intro
   --ad-hoc
   --copy-on-write
   --journaling

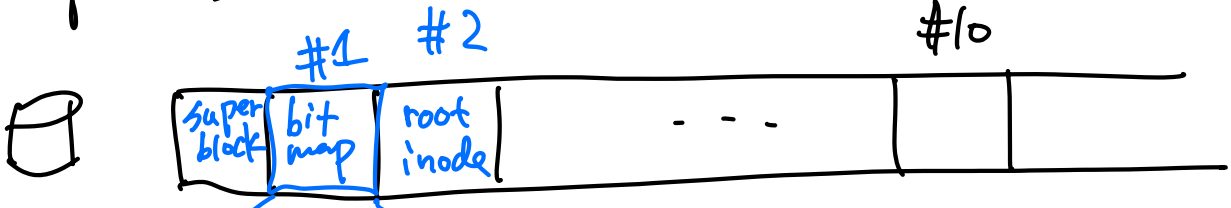
- Lab3 challenge

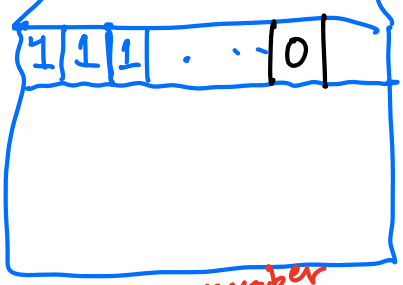
baseline: 50s
TA : ~25s

4

0.1s 500x

1. fs updates





4096*8 bits

0: free
1: used

RWX

4 2 1

1	0	0
1	1	0

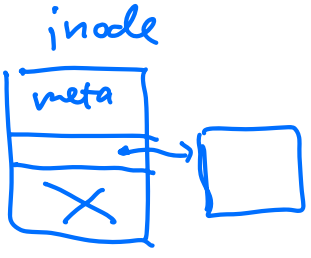
0x → hex oct number → 3bits

* mkdir("/dir1/", 0644)

Question: what does "0644" mean?

user: RW- R-- R-- world
group

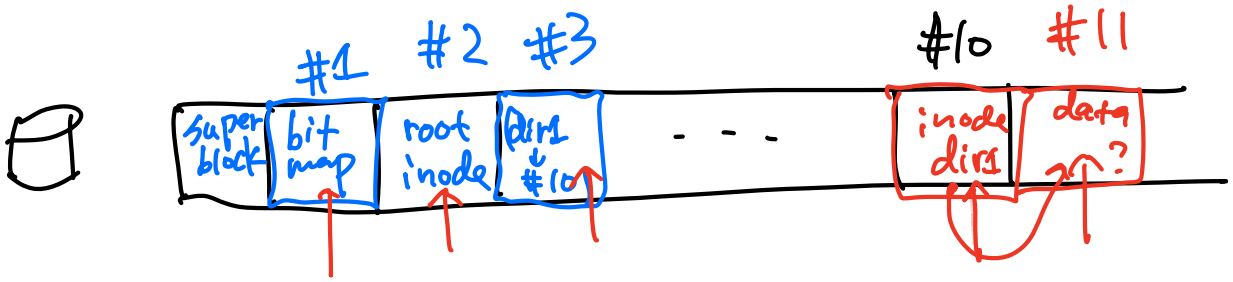
Question: how it works?



① Create inode for "dir1"
② update bitmap
③ fetch root's inode, fetch data block
④ add("dir1", #inode) → to data block
⑤ update root's metadata

Question: how many blocks will be written in this process?

2, 3, 4, 5



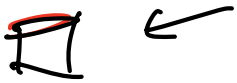
2. Crash consistency. "lost+found"

Crash #1 : lost 2 blocks

Crash #2 : garbage in #11

Crash #3 : "lost (2) blocks" standalone inode#10

Crash #4 : metadata inconsistency



atomicity

There are five writes in mkdir("/dir1", 0644):
(What can go wrong?)

Crash #1
1. block#1: bitmap, for allocating new blocks
2. block#10: create "dir1" inode

#2
3. block#11: init data block

#3
4. block#3: add direntry "dir1" to the parent dir ("/")

#4
5. block#2: update metadata of the parent dir

A ad-hoc fix (fsck)

B Cow fs (zfs)

C. Journaling (ext3, ext4)

goal: metadata consistency
data consistency

A. ad-hoc fix

fix #1: recycle #10, #11

fix#2, #3: "lost + found"

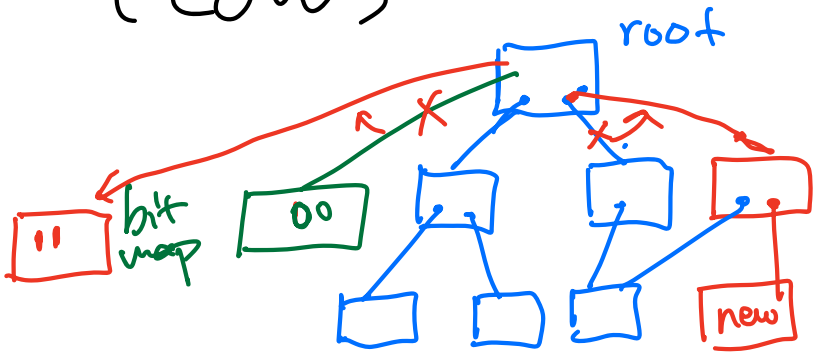
fix#4: ?

- unclear guarantees.
- ad-hoc reasoning what is right
- poor performance
- slow recovery

There are five writes in mkdir("/dir1", 0644):
(How to fix?)

fix #1
1. block#1: bitmap, for allocating new blocks
2. block#10: create "dir1" inode
#2
3. block#11: init data block
#3
4. block#3: add direntry "dir1" to the parent dir ("/")
#4
5. block#2: update metadata of the parent dir

B. Copy-on-Write approach (CoW)



+ commit in any order
+ always consistent
+ easy versioning
- write amplification
- space overhead

Q: zfs is almost FULL, deleting files is a good idea ?

NO!!

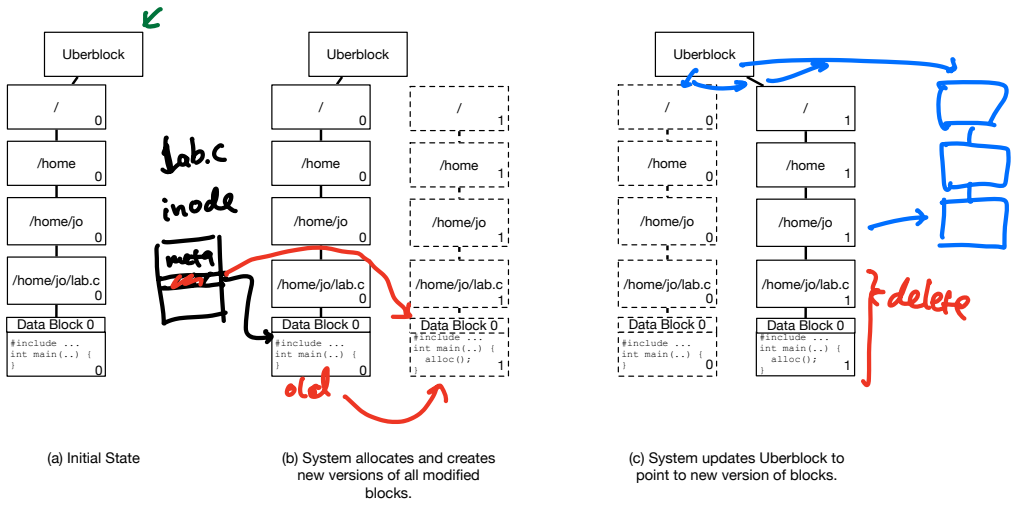


Figure 1: Copy-on-write filesystem: modifying a data block

Borrowed from NYU CS202 with minor updates:
https://cs.nyu.edu/~mwalfish/classes/21sp/lectures/handout13.pdf

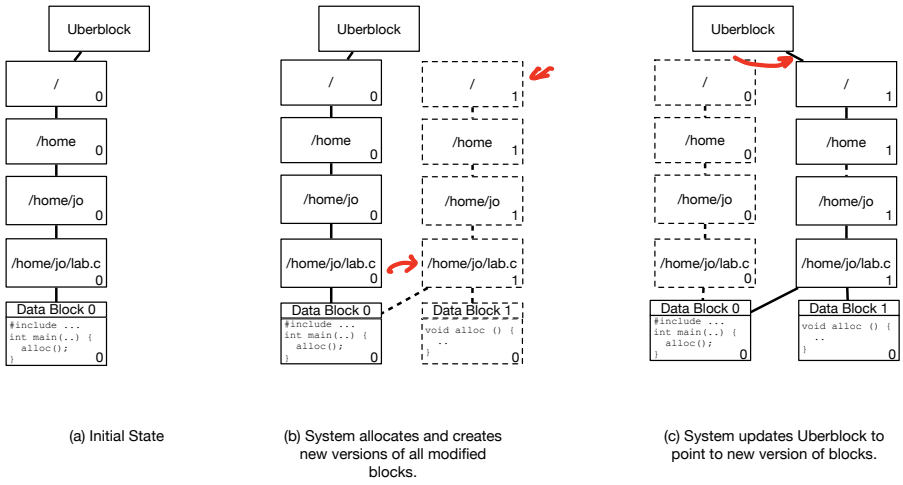
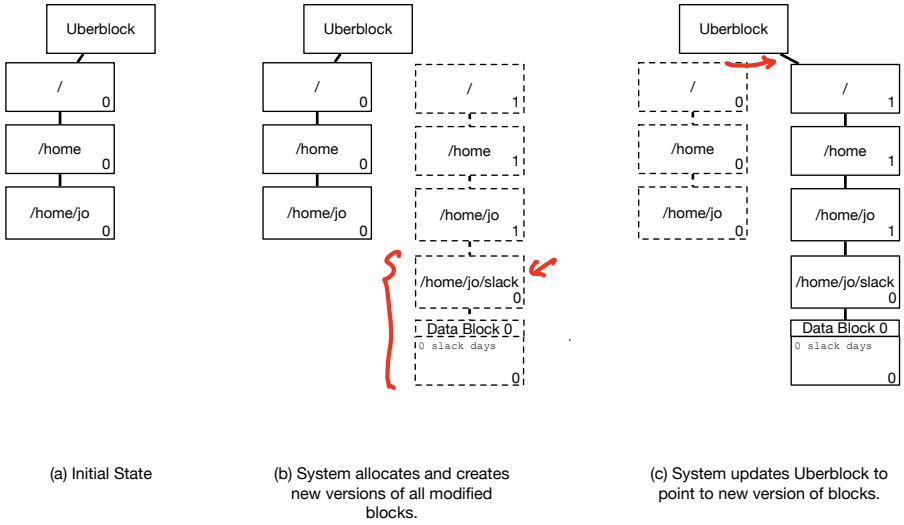


Figure 2: Copy-on-write filesystem: adding a data block



(a) Initial State

(b) System allocates and creates new versions of all modified blocks.

(c) System updates Uberblock to point to new version of blocks.

Figure 3: Copy-on-write filesystem: creating a file

ext3 disk layout

ext3 journal layout

Figure 4: Redo logging in a filesystem