CS202 (-002) Lecture 19: Fast file system, crash recovery, ...

Where we are

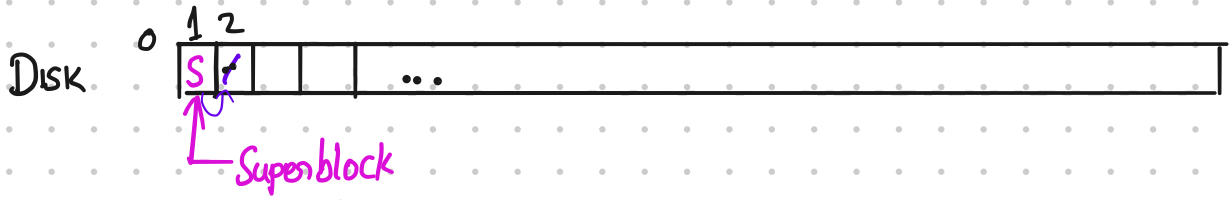
- File system <file, offset> → Location on disk
  Persistent mapping.
- Files

Inode (Identified by disk block number) → Metadata
→ Location of Data
↳ Tree of data blocks,
OR
→ Extents

- Directories
  - Special type of file
  - Data: Map from name → inode #
    - Links: hard / symbolic

Putting it all together

Logical



- At a known location
  ↳ So that OS can find it on boot
  → Metadata
  - Disk info
  - Free list ptrs, ...

/home/cs202/README.txt

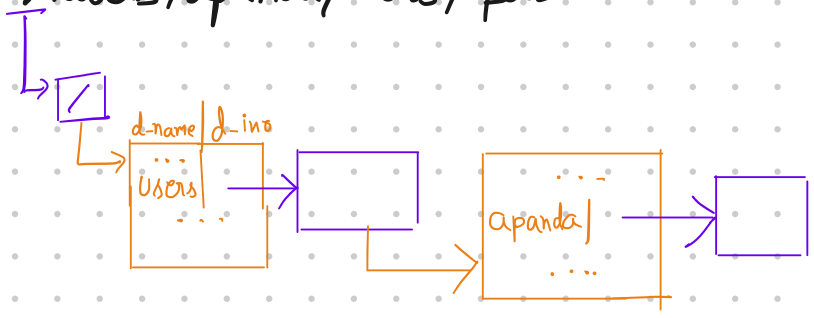
/user/cs202

/tmp/...

— Inode for /: Root directory

- /: Root of the file system
  - ↳ Can reach any path if this is known

/Users/apanda/class/pa25



→ The node for / is often inode 2.
ext4, etc.

Performance/Locality

Last Class →

Observations to help with meeting goals

ⓐ Often – all blocks in a file are accessed together
↳ Sequentially

ⓑ Most files are SMALL (a few disk blocks)

ⓒ Most of the DISK SPACE is used by large files

ⓓ Sequential access to disk faster than random access – Last lecture

Want file systems to be fast ⇒

Constraints on how disk blocks are

① Want a file's data to be in consecutive disk blocks to allow fast seq. access [a+d]

② Want inodes for files in a directory to be close to each other [d]
+ File metadata in a directory

ALLOCATED [

accessed together

Fast file system (FFS) – One design for this

[Still in use — OpenBSD]

- Core idea

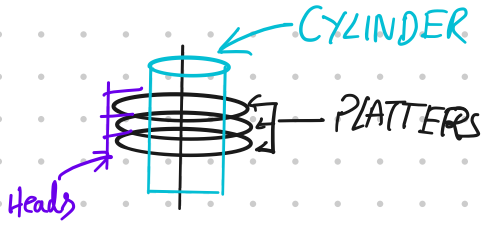
① Partition disk into groups of blocks that are CLOSE to each other

Distance metric : Time to access block B after block A

Disk geometry : Seek + Rotation

↑ Try to avoid

Cylinder Group



② Allocate files & directories so that COMMON access patterns access blocks in a single cylinder group.

DIRECTORY : File metadata accessed together

ⓐ New directory in cylinder group with largest # of FREE BLOCKS

ⓑ If possible, allocate inode for a new file in the same cylinder group as the directory containing it

FILE: Often, all blocks are accessed sequentially

↳ If possible

<48 KB: Same cylinder group as file's own inode

>48KB:

ⓐ When threshold is first crossed find cyl. group with largest # of free blocks

ⓑ Subsequently place blocks near each other

Observe: Need information about Cylinder Groups to implement: # of free blocks, etc.

Layout cylinder groups to help with this

CYLINDER GROUP (NOT Disk)

Track what blocks are available

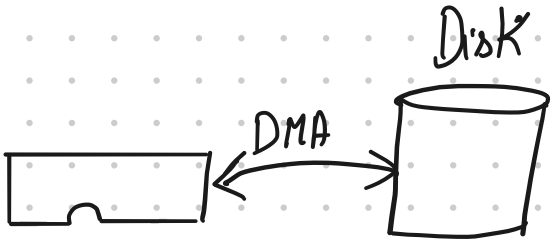
popcnt

11106...

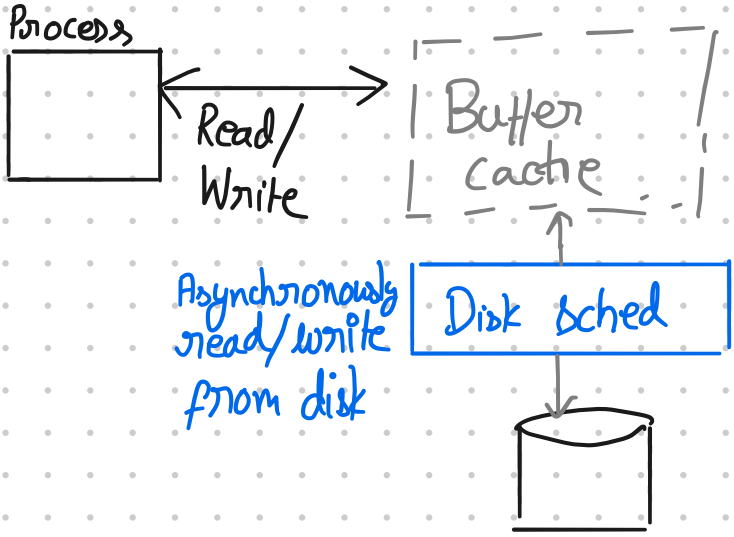
Use this in Lab 5.

Perf: Buffer cache

-



- Avoid reading blocks in if they are in memory
- Shared, OS managed memory cache
-



+ve: Better performance

-ve: Crash consistency

Crash consistency

- Problem:
  1. Buffer cache contains both metadata (inode, dir, etc.) & data
  2. Changes in buffer cache lost on

power failure or OS crash

- No dirent pointing to inode
- No inode pointing to data
- Inconsistent bitmap
- ...

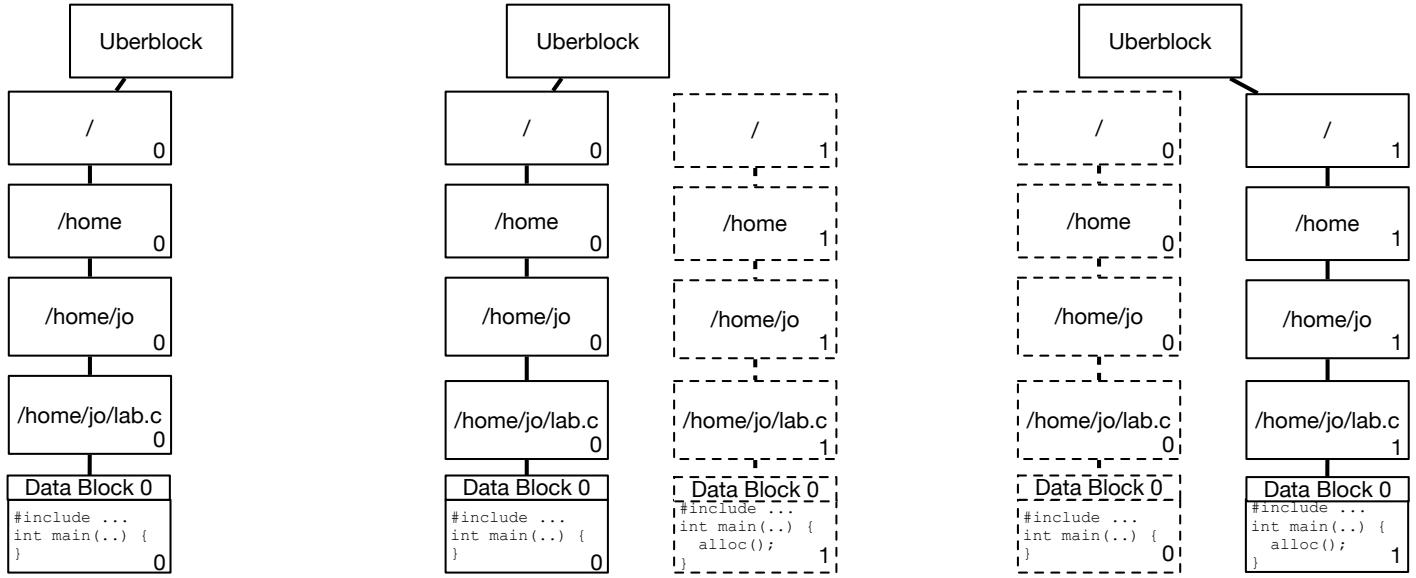
Desirable: Ensure file system is CONSISTENT after crash
- No dangling data blocks/inodes
- No inconsistent directories, etc.

Note: Cannot guarantee no data loss

Approaches

@ Copy-on-Write

CS202 Handout 11

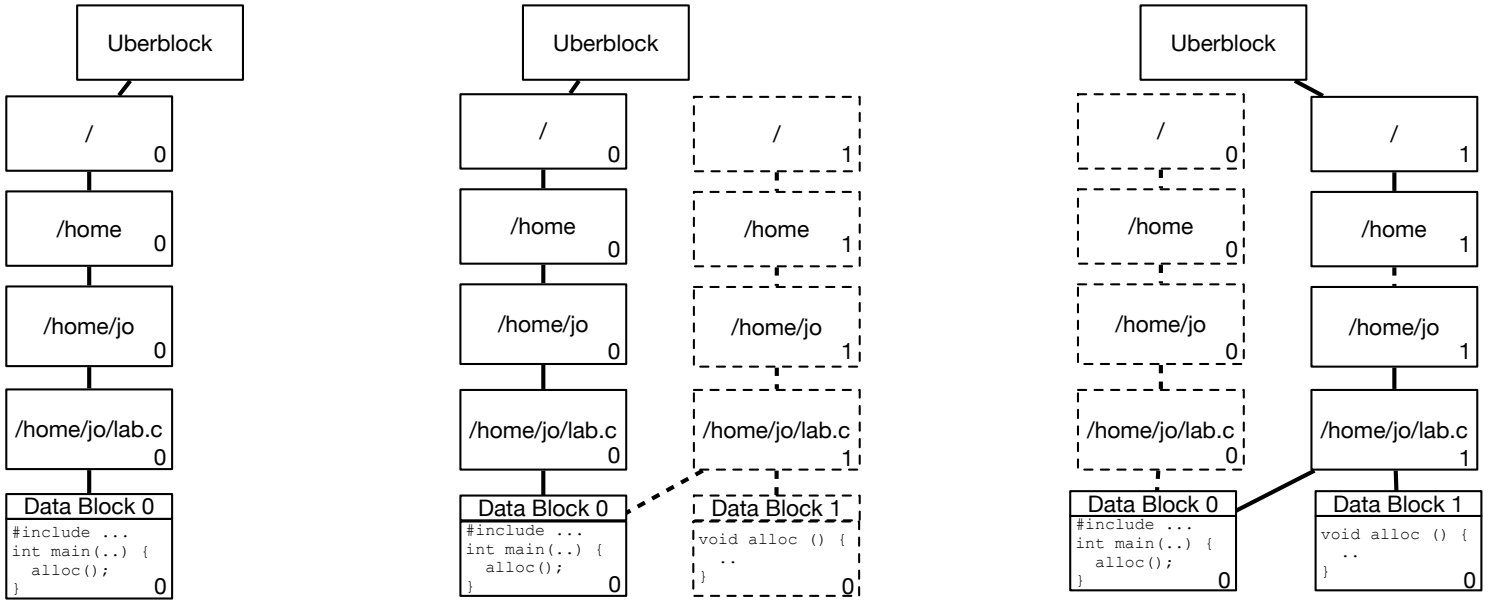


(a) Initial State

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

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

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

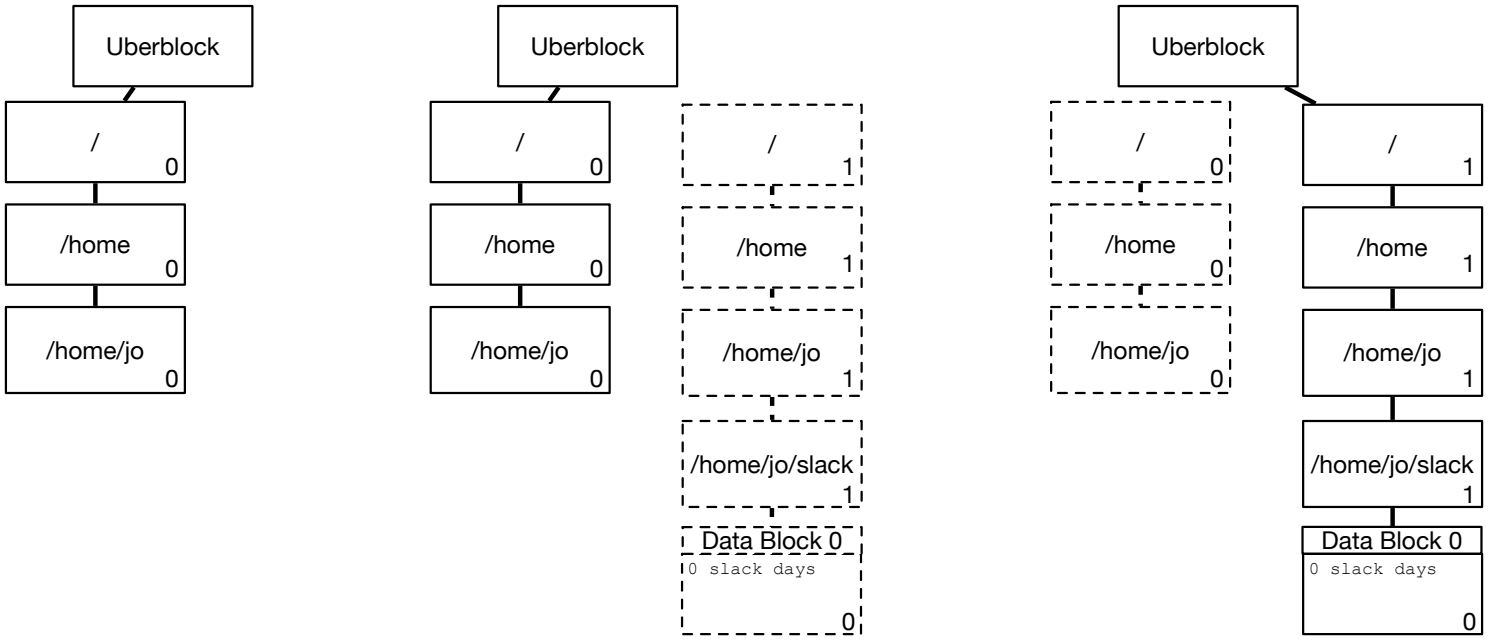


(a) Initial State

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

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

Figure 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

Figure 4: Redo logging in a filesystem