Locality and the Fast File System

Chapter 41

Previously in CS212…

- We looked at the basics of a file system implementation

- Disks are broken down into blocks

- We use part of the disk for holding data, and other portions for metadata
    - Inodes for block free list and file/directory location metadata
    - SUPERBLOCK for metadata about the filesystem

- However, we have a problem….

Poor Performance

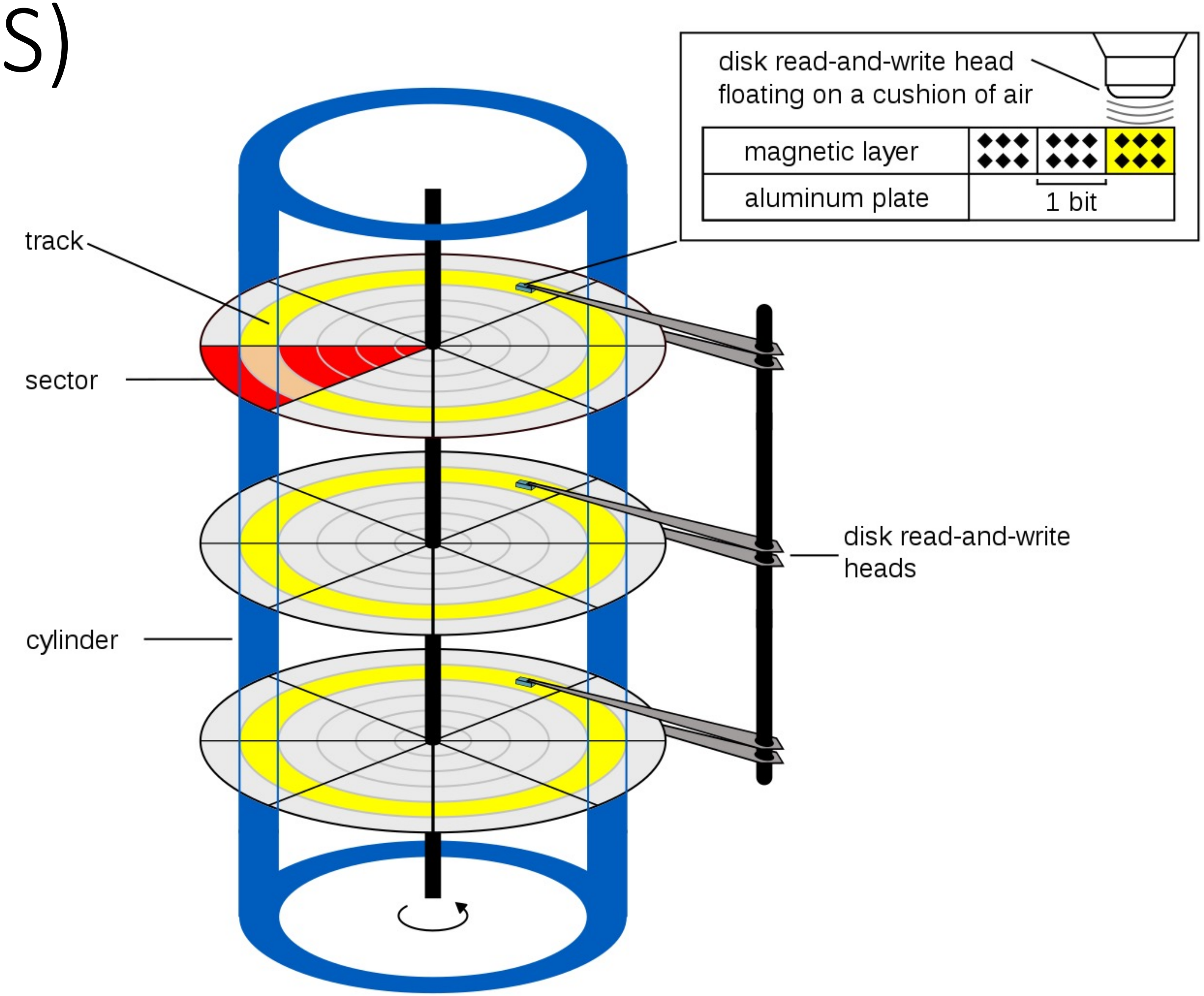
- The vsfs example and the old UNIX filesystem don’t store things with locality in mind

- Data is very far from the inodes which causes expensive seek operations

- Small block sizes minimized internal fragmentation
  - External fragmentation can cause files to be spread across multiple non-consecutive blocks
  - Mechanical HDDs benefit from defragmentation tools

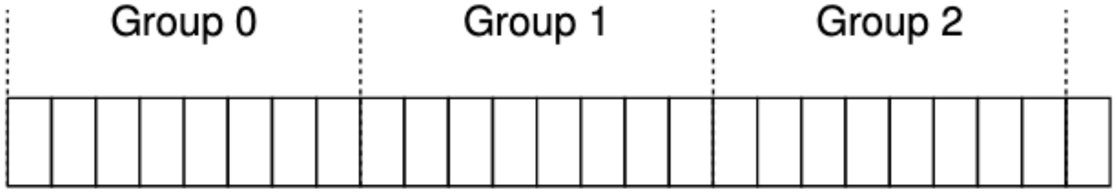
Fast File System (FFS)

- Built on top of the standard interface
  - open(), read(), write(), close()
- A **cylinder** is the set of tracks in the same position across all patter surfaces
- FFS collects consecutive cylinders into a **cylinder group**

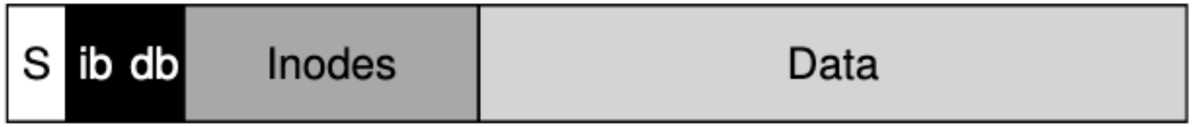


Logical Organization

- The HDD doesn’t share information about the geometry of the HDD
  - Only block addresses
- Modern file systems logically organize the drive into **block groups**
  - Each group is a consecutive portion of the disk address space



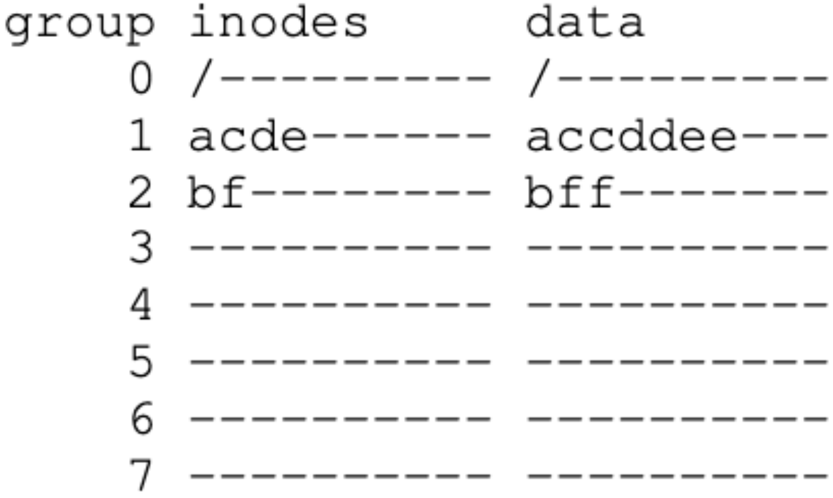
- The important thing is that data stored in the same group will not result in long seeking across the disk
- Each group keep track of its own file system structures



Allocating Files and Directories

- Keep related stuff together
  - Allocate data blocks for a file in the same group as its inode
  - Place all files that are in the same directory in the cylinder group of the directory they are in
- Assume we store /a/c, /a/d, /a/e, /b/f

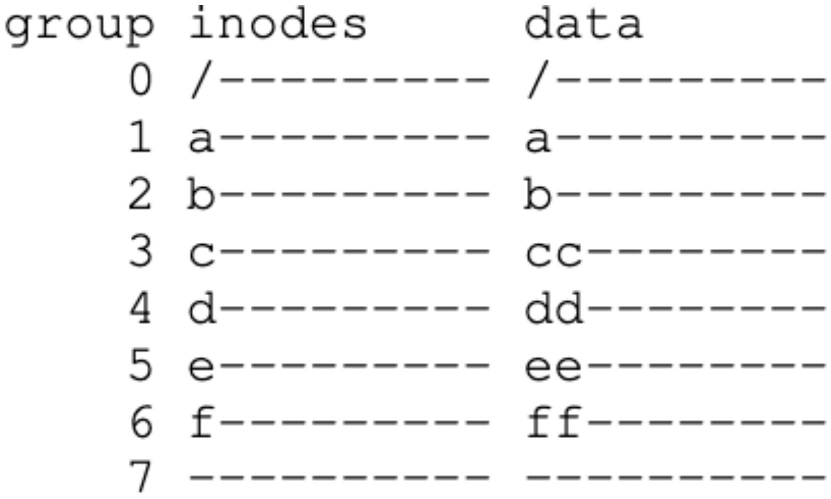
Cylinders with directory locality



```
group inodes     data
    0 /--------- /---------
    1 acde------ accddee---
    2 bf-------- bff-------
    3 ---------- ----------
    4 ---------- ----------
    5 ---------- ----------
    6 ---------- ----------
    7 ---------- ----------
```

Vs.

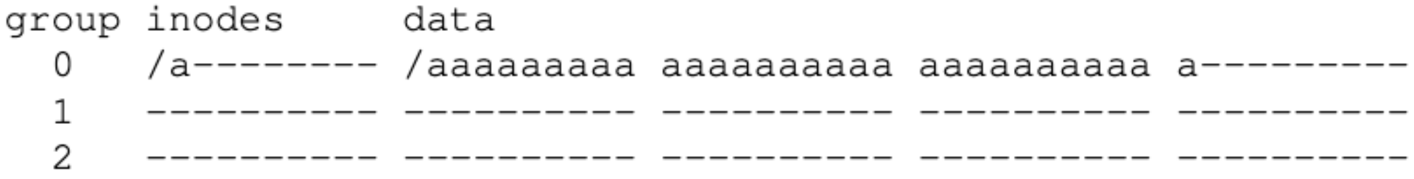
Cylinders without directory locality



```
group inodes     data
    0 /--------- /---------
    1 a--------- a---------
    2 b--------- b---------
    3 c--------- cc--------
    4 d--------- dd--------
    5 e--------- ee--------
    6 f--------- ff--------
    7 ---------- ----------
```

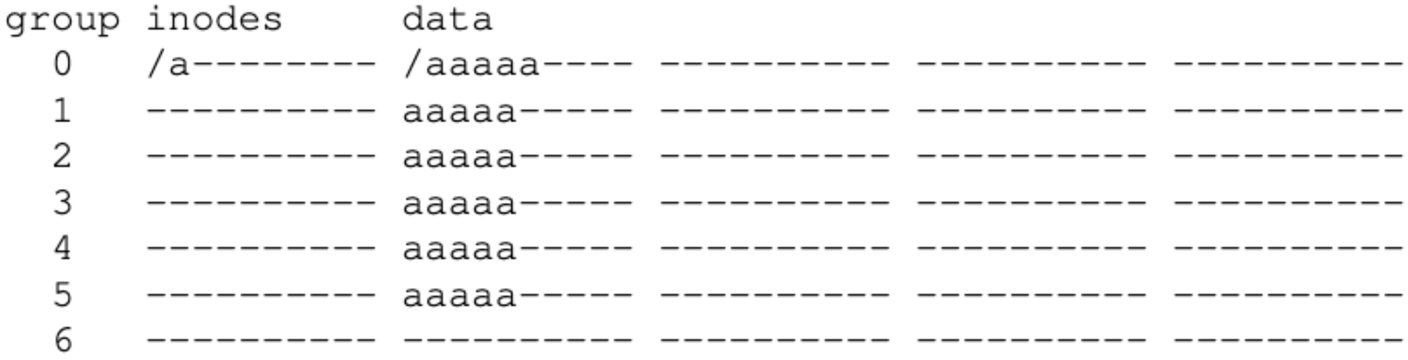
Large-File Exception

- A large file could take up all the space in a group
  - This means we might not be able to store other files in the same directory within the same group



```
group inodes     data
  0   /a-------- /aaaaaaaaa aaaaaaaaaa aaaaaaaaaa a---------
  1   ---------- ---------- ---------- ---------- ----------
  2   ---------- ---------- ---------- ---------- ----------
```

- We instead use up the direct blocks first and use the indirect pointers to store the remaining content in other block groups
  - Likely with less utilization



```
group inodes     data
  0   /a-------- /aaaaa---- ---------- ---------- ----------
  1   ---------- aaaaa----- ---------- ---------- ----------
  2   ---------- aaaaa----- ---------- ---------- ----------
  3   ---------- aaaaa----- ---------- ---------- ----------
  4   ---------- aaaaa----- ---------- ---------- ----------
  5   ---------- aaaaa----- ---------- ---------- ----------
  6   ---------- ---------- ---------- ---------- ----------
```

- Does have performance issues, but we can limit it with larger chunks
  - A chuck is just a unit of how much data we read/write from the disk

Other FFS Features

- Sub-blocks to hold small files
    - Reduce internal fragmentation
    - Mostly avoided by having the library buffer the data and write out when it’s large enough
- Parameterization
    - Skip blocks when writing to consecutive data to about “missing” data on a rotation
    - Modern drives have a track buffer to hold a track and reach from this cache on subsequent reads for that track

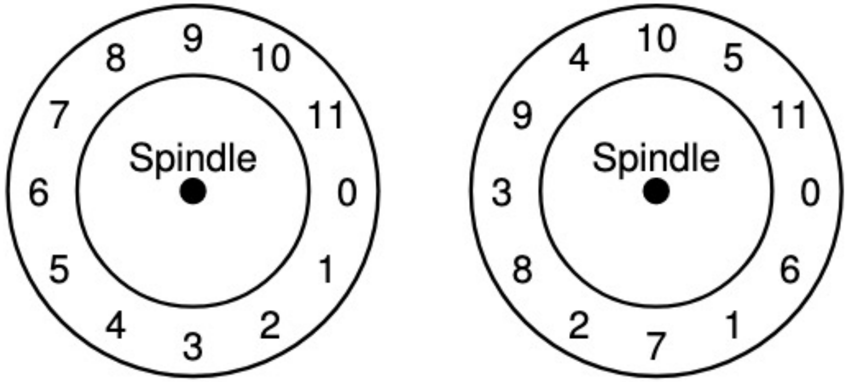


Figure 41.3: **FFS: Standard Versus Parameterized Placement**

Next Time

- We talk about how we handle situations when things go wrong…