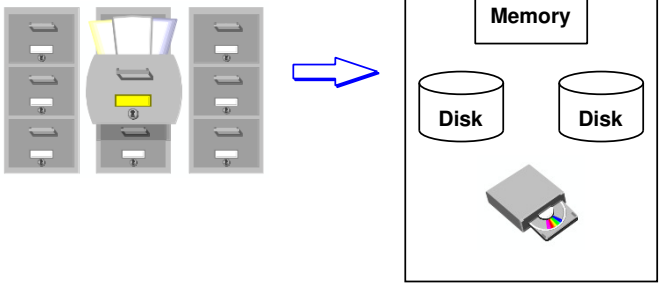
Ch 6: File Systems

Bill Cheng

http://merlot.usc.edu/cs402-s16

Files



Requirements

- Permanent storage
  - resides on disk (or alternatives)
  - survives software and hardware crashes
    - (including loss of disk?)
- Quick, easy, and efficient
  - satisfies needs of most applications
    - how do applications use permanent storage?

Applications

- Software development
  - text editors
  - linkers and loaders
  - source-code control
- Document processing
  - editing
  - browsing
- Web stuff
  - serving
  - browsing
- Program execution
  - paging

Needs

- Directories
  - convenient naming
  - fast lookup
- File access
  - sequential is very common!
  - "random access" is relatively rare

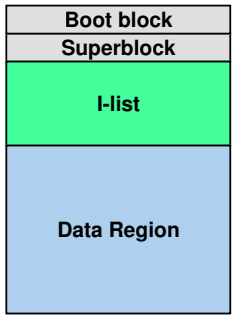
6.1 The Basics of File Systems

- *UNIX's S5FS*
- Disk Architecture
- Problems with S5FS
- Improving Performance

S5FS

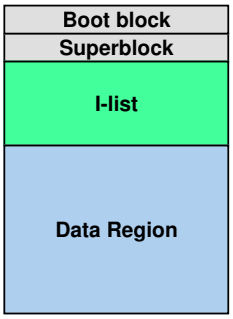
- A simple file system
  - slow
  - not terribly tolerant to crashes
  - reasonably efficient in space
    - no compression
- Concerns
  - on-disk data structures
    - file representation
    - free space

S5FS Layout



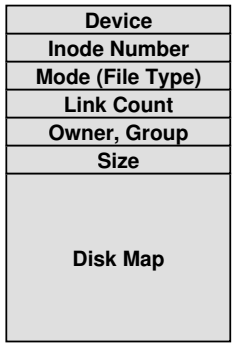
- A disk is simply an array of blocks of 1KB each (old Unix: 512B)
- A "linear view" (1-D array of blocks) of the disk

S5FS Layout



- The *superblock*
  - describes the layout of the rest of the file system
  - contains the *head* of the *free list*
- The *i-list* is an *array* of *index nodes (inodes)*
  - each representing a *file*

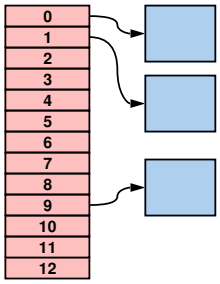
S5FS: Inode



Disk Map

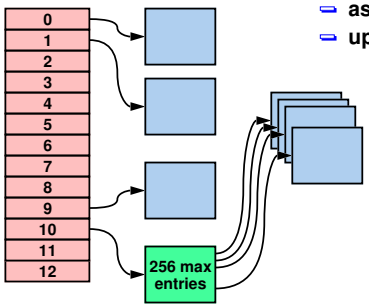
- assuming blocksize = 1KB

Disk Map



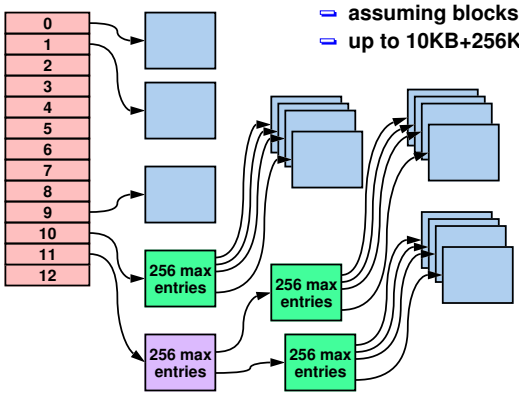
- assuming blocksize = 1KB
- up to 10KB

Disk Map



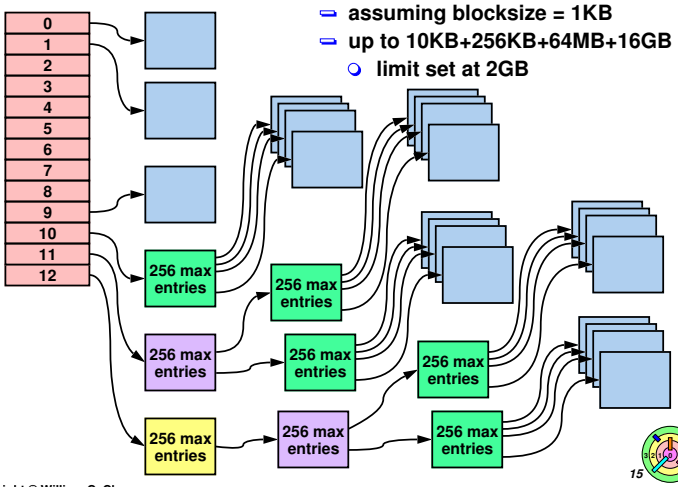
- assuming blocksize = 1KB
- up to 10KB+256KB

Disk Map



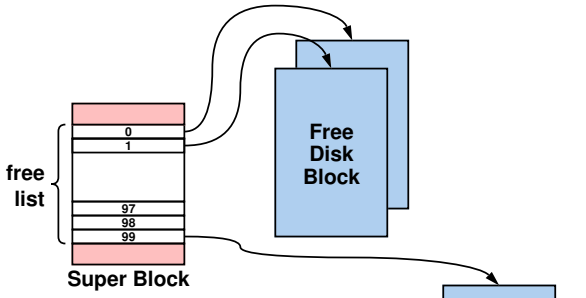
- assuming blocksize = 1KB
- up to 10KB+256KB+64MB

Disk Map



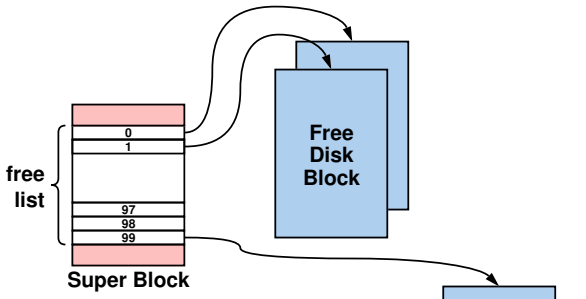
- assuming blocksize = 1KB
- up to 10KB+256KB+64MB+16GB
  - limit set at 2GB

S5FS Free List



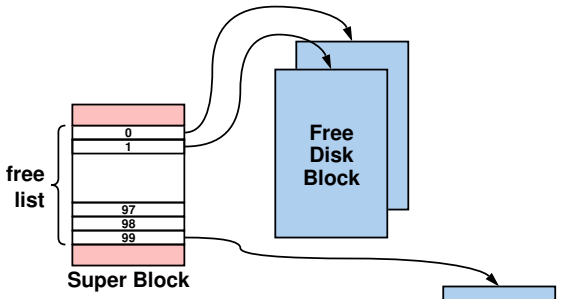
- The superblock contains the addresses of up to 100 *free disk blocks*
  - the last of these disk blocks contains 100 pointers to additional free disk blocks, etc.
  - can find *all* the free disk blocks this way

S5FS Free List



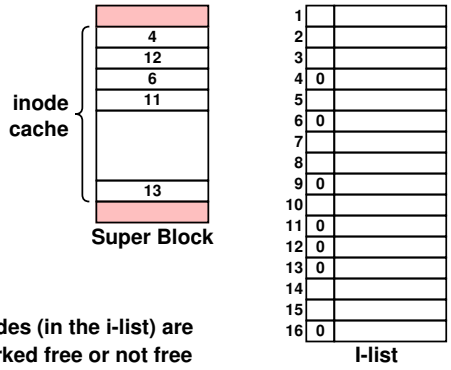
- If the last free disk block in the superblock is used
  - copy 100 pointers into the superblock with one disk access

S5FS Free List



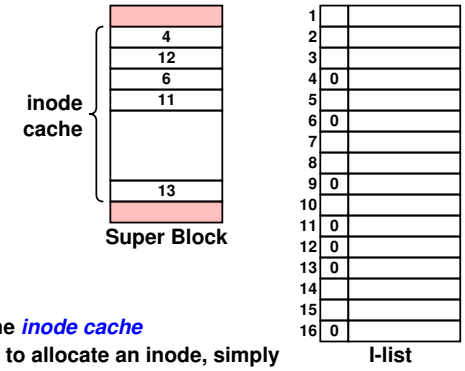
- When a disk block is freed
  - that block's address is added to the list of free blocks in the superblock
  - if the list is full
    - write it out and update superblock

S5FS Free Inode List



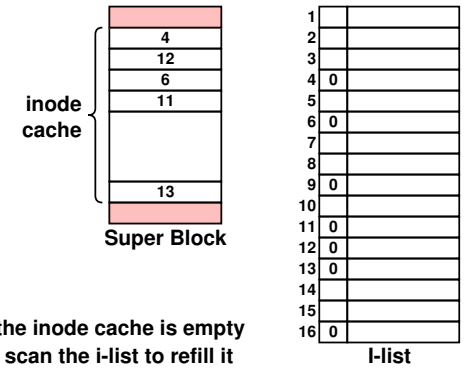
- Inodes (in the i-list) are marked free or not free
  - no additional organization in the i-list
  - the superblock *caches* free inodes (i.e., in the *inode cache*)

S5FS Free Inode List



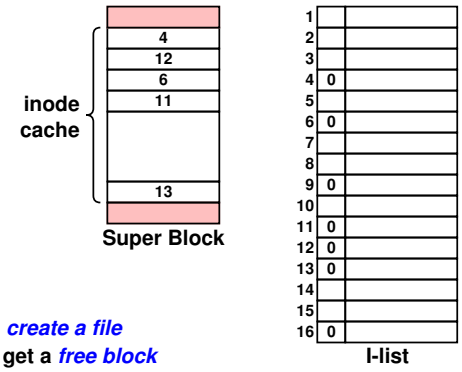
- The *inode cache*
  - to allocate an inode, simply mark it not free and remove it from the inode cache
  - to free an inode, simply mark it free and add to the inode cache if there is room

S5FS Free Inode List



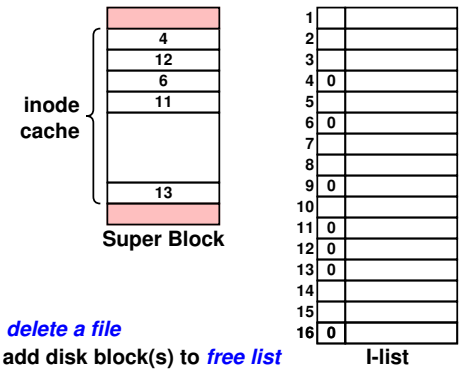
- If the inode cache is empty
  - scan the i-list to refill it
    - to help out with the scan, the inode cache contains the index of the first free inode in the i-list
      - need to maintain this entry when necessary

S5FS Free Inode List



- To *create a file*
  - get a *free block*
    - update *free list*
  - get a *free inode*
    - update *i-list* and *inode cache*

S5FS Free Inode List



- To *delete a file*
  - add disk block(s) to *free list*
  - mark inode free in *i-list* and may be update *inode cache*

S5FS Summary

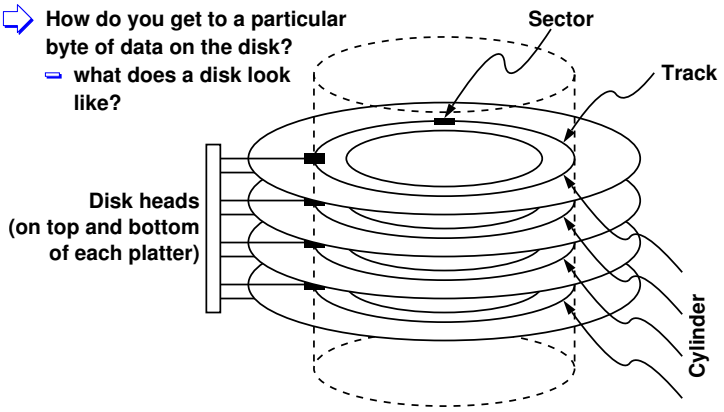
- In designing a file system, one tries to minimize the number of disk operations
  - read vs. write
  - sequential access vs. random access
    - S5FS gives O(1) number of disk operations for random access

6.1 The Basics of File Systems

- UNIX's S5FS
- *Disk Architecture*
- Problems with S5FS
- Improving Performance

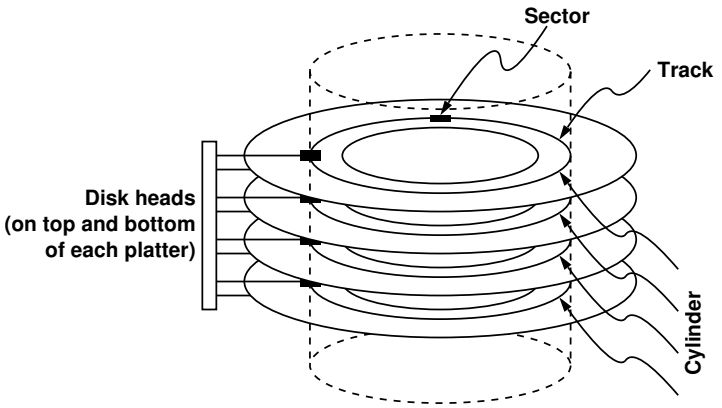
Disk Architecture

- How do you get to a particular byte of data on the disk?
  - what does a disk look like?



- Clearly, this looks nothing like the S5FS layout
  - or any logical file system layout

Disk Architecture



- Smallest addressable unit is a *sector*
  - disk address = *(head/surface#, cylinder/track#, sector#)*

Rhinopias Disk Drive

Rotation speed	10,000 RPM
Number of surfaces	8
Sector size	512 bytes
Sectors/track	500-1000; 750 average
Tracks/surface	100,000
Storage capacity	307.2 billion bytes
Average seek time	4 milliseconds
One-track seek time	.2 milliseconds
Maximum seek time	10 milliseconds