UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 537
Intro to Operating Systems

A. Arpaci-Dusseau
Spring 2000

Current File System Issues?

Questions answered in these notes:

- What problems weren't solved in FFS?
- What are RAIDs?
- What is the goal of a Log-structured file system (LFS)?
- How are multiple file systems supported in UNIX?

Readings for File System Topic: Chapters 10 - 13

Limitations of FFS

Block-based rather than extent-based

- Separate disk request for every block --> Fixed overhead
- Optimization (Linux **ext2fs**)
  - Cluster contiguous blocks with single pointer and length

Crash recovery

- Writes of data blocks performed asynchronously
  - Improves performance versus synchronous writes
  - Flush to disk every 30 seconds
- Meta-data written synchronously
- Check file system: fsck

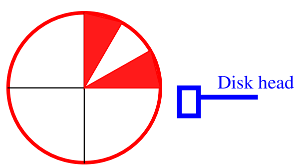
Small file performance (most files are small)

- Synchronous meta-data hurts performance

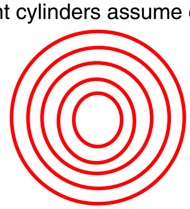
FFS Limitations Continued

Dependence on disk geometry

- Move disk from faster machine to slower



- Does not handle multi-zone disks
  - Calculations for adjacent cylinders assume constant number of sectors



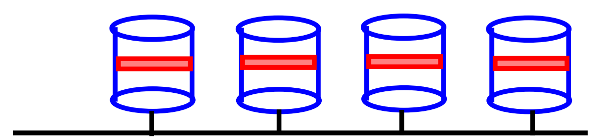
- Does not handle disk arrays (RAIDs)

RAIDs

Disk manufacturers: Focus on making small disks well

- Combine multiple small disks to provide capacity and performance

Idea: Stripe data from file system across multiple disks



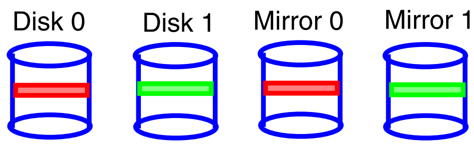
Problem: More disks increases probability that any one will fail

- Metric: Mean Time To Failure (MTTF)
- Solution: Add redundancy
- Multiple copies of data across different disks

Redundant Array of Inexpensive Disks

Adding Redundancy to RAIDs

Mirror (shadow) data: Write all data to two disks



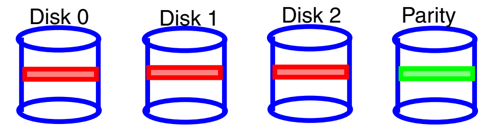
Advantages

- Very simple to implement
- Tolerate failure of any one disk: Just read from mirror
- Improve bandwidth of reads

Disadvantage: Expensive

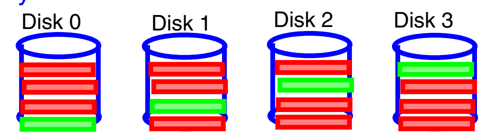
Parity in RAIDs

Use one disk for parity (XOR each bit across each block)



- Advantage: Wastes less space for redundancy than mirroring
- Disadvantage: Must recalculate parity on every write
  - Read block from every disk
  - Write data and parity disk

Interleave parity blocks across all disks



- Advantage: Evenly distribute activity across disks

Log-Structured File System

Technology motivation

- Faster CPUs: I/O becomes more and more of a bottleneck
- More memory: File caches are effective for reads
  - Implication: **Writes** compose most of disk traffic
- Popularity of RAIDs:
  - Opportunity to leverage concurrency
  - Increases cost of small writes

Problems with previous file systems

- Perform many small writes
  - Good performance on large, sequential files, but most files are small
- Synchronous operation
  - Cannot leverage concurrency of RAID
- Depends upon knowledge of disk geometry

Idea of LFS

Insight: Treat disk like a tape-drive

- Best performance from disk for sequential operations

Write data to disk in a sequential log

- Delay all write operations
- Write meta-data and data for all files intermixed in one operation
- Do not overwrite old data on disk

Advantages

- Large sequential write operations --> good performance
- No knowledge of disk geometry
  - Assume sequential better than random

Potential Problems

- How do you find data to read?
- What happens when you fill up the disk?

Reading in LFS

Same basic structures as UNIX

directories
inodes
indirect blocks
data blocks

- Retrieval of data block implies finding the file's inode
- UNIX: Inodes kept in array (per partition or per cylinder group)
- LFS: Inodes move around on disk

Solution: Inode map indicates where each inode is stored

- Inode map small enough to keep in memory
- Inode map written to log with everything else
- Written to known checkpoint location on disk for crash recovery

Disk Cleaning

Disk runs low on free space

- Run a disk cleaning process
- Compacts live information to contiguous blocks of disk

Problem: Long-lived information repeatedly copied over time

- Solution: Partition disk into segments
  - Group older files into same segment
  - Do not clean segments with old files

Try to run cleaner when disk is not being utilized

LFS: Neat idea, used in Sprite operating system

- Continues to be an active area of research
- Use in distributed environments

Supporting Multiple File System

Want to be able to support multiple file systems

- Local file system (FFS, LFS, fifoFS)
- Distributed file system (NFS, AFS)
- /proc

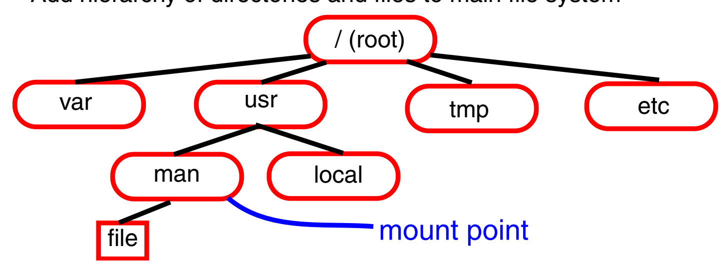
Add file system independent layer

- Data abstraction: **vnode** (virtual node)
- Kernel's generic abstraction of a file
- vnode contains pointer to data structure dependent on file system

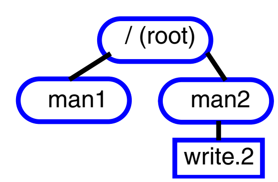
Multiple File Systems in Hierarchy

Mountable file systems

- Add hierarchy of directories and files to main file system



- Sub File System



Mounting in UNIX

Mounted file system transparent to users

- Only super-user can mount new file systems (mount + umount)
- Verify that file system being mounted is valid
- Add mount point and root of new file system to **mount table**
- Set special bit in inode

Reference to mounted file system

- See bit set in inode
- Search mount table for inode of mounted file system

/proc File System (procfs)

File system interface to processes in system

- /proc: mount point
- Open, read, and write information about any process (w/ permissions)

  read(), write() and ioctl() interface

  ls -l /proc
  -rw------- 1 root root 27680 Nov 17 15:31 00001 PID
  -rw------- 1 root root 1853Nov 17 15:31 00378
  -rw------- 1 root root 3276Nov 17 15:31 00853
  -rw------- 1 ad1dgrad9530 Nov 17 19:59 01059

/proc leveraged by many system utilities

ps, debugger