Outline

- Where have we been
- Log structured file system
- What next (week)

Where have we been

- Abstractions to create a container for a running program (process), mechanisms to assign CPU resources (threads), mechanisms to ensure that processes can share/run simultaneously (locks, semaphores, deadlocks) and use CPU resources “fairly” (CPU scheduling)
  - Homework assignment
- File systems: Overview, typical file system access patterns, ffs (attempt to use disk characteristics to build better file system)
- Today: log structured file system (another attempt to improve fs performance)
- ‘morrow: RAID - using parallelism
- Distributed fs…

Project wise

- We've installed a new kernel, configured it appropriately for the current machine, tinkered with something and see if what we expect is what happens
- We will continue with that model for the file system before trying something with threads
- Today: Summary: Boehnen, Pro: Durnan, Con: Ryan

Log structured file system

- Modern machines have lots of memory. This memory can be used to catch most of the (repeated) reads. In fact, if you have a laptop/desktop with 2 GB main memory you might have all your popular read-only data in cache
- Writes cannot be delayed indefinitely because of data safety
  - Most file operations are writes and prefetches
  - Most files are small
  - Writing a small file can take five disk operations, read data/index for directory entry and read/write of inode entry (update meta information such are last modified) before writing actual data = 5% utilization

lfs

- You collect all writes and write it all at once to disk, paying for a single seek
- File systems such as NTFS, ext3 etc. use logs to store latest data (that are eventually moved into the non-log areas). This paper talks about a system that only contains logs; there are no “other” areas
- They show that for small writes, they are significantly better, for other cases they are similar to ffs

Technical challenges

- Need large contiguous space
  - Lfs uses segments to compartmentalize free space
- Checkpoints, written into fixed region keeps track of inode maps
- Inode map keeps track of inodes - kept in memory
- Inodes are written into logs
- Free space management
  - Thread segments
  - Copy live data into new segments for cleaning
  - Long lived data might get moved around multiple times as data around them get deleted

Technical challenges

- Its more important to clean cold segments than hot segments
- Cleaning can also be used to realign data (to be closer/sequential)
- Crash recovery
  - Checkpoint - two phase protocol
  - Roll-forward - play back logged meta information since last successful checkpoint

‘morrow: RAID

- Disks are improving tremendously in capacity and size but not in performance
  - Disks have mechanical components: we’ve been stuck in 15K RPM for a while now, seeks are not getting faster either
  - With large capacity come reliability problems
    - Backups have to keepup
  - RAID: Use redundancy (disks are cheap) to achieve reliability, use redundancy to parallelize disk I/O for more performance, use cheap disks (instead of top of the line disks) to offset the cost of redundancy

Resources

- http://www.acnc.com/raid.html - cute icons and explanation