CSE 451: Operating Systems
Winter 2006

Module 16
Journaling File Systems

Ed Lazowska
lazowska@cs.washington.edu
Allen Center 570

In our most recent exciting episodes …

- Original Bell Labs UNIX file system
  - a *simple yet practical* design
  - exemplifies engineering tradeoffs that are pervasive in system design
  - elegant but slow
    - and performance gets worse as disks get larger
- BSD UNIX Fast File System (FFS)
  - solves the throughput problem
    - larger blocks
    - cylinder groups
    - awareness of disk performance details

Both are real dogs when a crash occurs

- Buffering is necessary for performance
- So the disk itself may be in an inconsistent state
  - metadata updated but data not
  - data updated but metadata not
  - either or both partially updated
- Fsck (i-check, d-check) are *very* slow
  - must touch every block
  - worse as disks get larger!

Journaling file systems

- Became popular ~2002
- There are several options that differ in their details
  - Ext3, ReiserFS, XFS, JFS
- Basic idea
  - update metadata, or all data, *transactionally*
    - *"all or nothing"*
  - if a crash occurs, you may lose a bit of work, but the disk will be in a consistent state
    - more precisely, you will be able to quickly get it to a consistent state by using the transaction log/journal – rather than scanning every frigging block

Undo log

- Log: an append-only file containing log records
  - <start t>
    - transaction t has begin
  - <t,x,v>
    - transaction t has updated block x and its old value was v
  - <commit t>
    - transaction t has committed
- Rules
  - if t modifies x, <t,x,v> must be written to disk before the updated x is written to disk
  - if t commits, then all modified blocks x must be written to disk before <commit t>

If a crash occurs

- Recover the log
- Committed transactions are problem-free
  - all data was updated on disk before the commit record was written
- Uncommitted transactions
  - The log contains the old value of each block involved in the transaction
  - By going through the log, you can restore the disk to the (consistent) state before the transaction was begun
- Note that once a transaction has committed, it can be deleted from the log

Impact on performance

- The log is a big contiguous write
  - very efficient
- And you do fewer synchronous writes
  - very costly in terms of performance
- So journaling file systems can actually improve performance (immensely)
- As well as making recovery very efficient

Want to know more?

- CSE 444! This is a direct ripoff of database system techniques
  - But it is *not* Microsoft Windows Longhorn – “the file system is a database”
  - Nor is it a “log-structured file system” – there is no file system, just a log