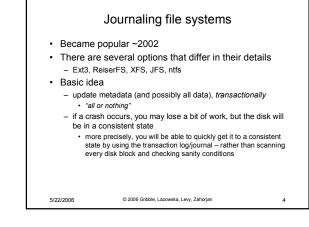
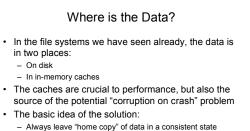


## Both are real dogs when a crash occurs Buffering is necessary for performance Suppose a crash occurs during a file creation: 1. Allocate a free inode 2. Point directory entry at the new inode In general, after a crash the disk data structures may be in an inconsistent state - metadata updated but 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! © 2006 Gribble, Lazowska, Levy, Zahorjan 5/22/2006 3





- Always leave "nome copy" of data in a consistent state
  Make updates persistent by writing them to a sequential (chronological) journal partition/file
- At your leisure, push the updates (in order) to the home copies and reclaim the journal space

5/22/2006

© 2006 Gribble, Lazowska, Levy, Zahorjan

5

## Redo log · Log: an append-only file containing log records – <start t> transaction t has begun - <t x v> · transaction t has updated block x and its new value is v Can log block "diffs" instead of full blocks – <commit t> · transaction t has committed - updates will survive a crash Comments - Committing involves writing the redo records - the home data needn't be updated at this time - No guarantees about the consistency of the file data, as seen by the application. (This is just how things were to begin with...) 5/22/2006 © 2006 Gribble, Lazowska, Levy, Zahorjan 6

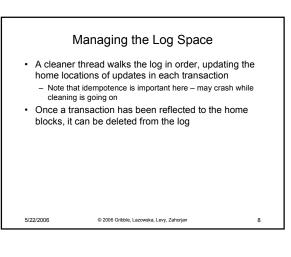


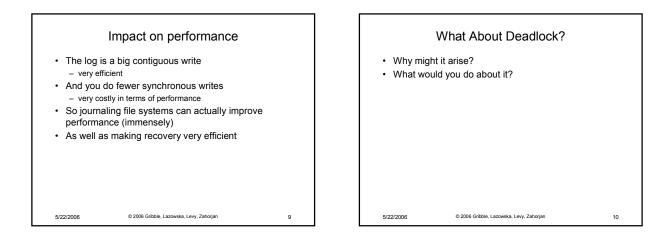
· Recover the log

5/22/2006

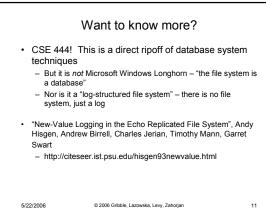
- · Redo committed transactions
  - Walk the log in order and re-execute updates from all committed transactions
  - Aside: note that update (write) is *idempotent*: can be done any positive number of times with the same result.
- Uncommitted transactions
  - Ignore them. It's as though the crash occurred a tiny bit earlier...

© 2006 Gribble, Lazowska, Levy, Zahorjan





7



11