Reimplementing the Cedar File System Using Logging and Group Commit
Robert Hagmann
SOSP 1987

Why are we reading this paper?
  Robustness and crash recovery
  Logging will come up many times

What are the main properties the authors want?
  Recoverable if unexpected crash (e.g. while modifying name table).
  If single- or double-sector disk error, only that file damaged.
  Fast crash recovery.
  Fast ordinary operation.

Basic on-disk FSD data structures
  name table, a b-tree, contains file names and block lists
  content blocks of files

What data structures should look like after a file create
  new blocks for file contents
  new entry in file name table

Why unexpected crashes might be a problem
  b-tree update requires multiple disk writes

Why does a log help with crash recovery?
  It makes multi-step operations atomic w.r.t. recovery

What does the on-disk log look like?
  Fixed area of disk, not a file, no free list for log blocks.
  Proper entry: hdr, blank, hdr copy, data pages..., end, copies of data..., end copy.

be clear about memory vs disk
  memory:
    disk cache. some blocks might be dirty.
    in-memory log, just most recent, not yet flushed, w/ commit points.
    VAM
  disk:
    real homes of data and name table blocks.
    on-disk log, in thirds.

What happens when you create a file in FSD?
  1. get two free pages for leader and data from VAM.
  2. update the name table b-tree in memory
  3. write leader+data to disk synchronously
  4. append modified name table pages to the log in memory

When is the in-memory log forced to the disk?
  Group commit... Every few seconds.
  Why not log immediately, after every operation?

When are the modified disk cache pages written to the disk?
  Is it OK to write them before the corresponding log records?
  Is it OK to write them after?

What happens during recovery after crash+reboot?

```
  How does recovery s/w find the start of the log?
    Disk has pointer to first log of oldest third, updated on entry to
third.
  How does recovery find the last log record?
    hdr/end blocks must have time-stamp or something.
  What if the system crashed while writing to the log on disk?
    Do we assume disk wrote pages of log entry in order?
    If end block exists and matches hdr block, we're OK.
  What does recovery actually do with the log records?
  Can it just start re-play at the beginning?
    I.e. what if it crashed during previous replay, is 2nd time OK?
  Where does recovery stop re-playing?

Does recovery work if crash during log write?

What about recovering the free list (VAM)?
  What if the free list had incorrect contents?

What can we say about the state of the file system after
crash+recovery?
  Some prefix of the operations have been completed.
  I.e. every operation up to a certain point.
  May have lost the last few operations.

Why don't they log data writes?
  When do they write the data?
  Does that break the prefix rule?
  May have writes from "the future", after the recovery point.
  What about delete, create, write, crash? W/ recovery point before
delete?

What if we run out of log space during normal operation?
  How can we be sure it's safe to start re-using the log space on disk?
  What if crash while writing blocks during switch to a new 1/3?

Why is FSD faster than CFS?
  Table 3, why does FSD do 1/6th as many I/Os for small creates?
  Why 149 I/Os for 100 files?

Did they evaluate whether they achieved their robustness goals?
```