LFS - OVERVIEW

In contrast, to update a file in this system:

- The log sequential write is just one second.
- Write the write map to the log.
- Write the data to the end of the log, then a log.
- Assume (for now) that disk size is infinite. To create a file, LFS checks

LFS: LOG-Structured Filesystem

Performance:

- To make write operations sequential, and hence
- LFS addresses

Recovery:

- Go back to a consistent state on crash.
- Arithmetic: to maintain complete record of every operation.

Stability:

- As backup copy for primary storage.

In general, why are logs used? Recall from Chapter 5:

LFS - USES OF LOGS

FREE SPACE MANAGEMENT

4

5
Recovery: Challenges

- Performance: Clean, the memory and networks are usually slow, so we use caches
- Why?
  - RAID: Replacing disk data on other disks
- Network: Network caches replace networked data on the local
- Disk: Replace disk data in memory
  - 1/1 and 1/2 processor caches replacing RAID data on chip
- When and why we need to keep multiple copies of data at different sites

We often have to keep multiple copies of data at different sites

**Recovery and Consistency**

**How to Group Blocks?**

\[ \text{cost} = \frac{1}{n + 1} \times \frac{\text{size of data}}{\text{size of data}} \]

**Why?**

- When segments to clean? Defer, cost too high
- How long to execute? Until 50–100 segments are free
- When to execute? When only a few tens of segments are free

**LFS — Cleaning Policies**

**Direct Operation Log to XXX**

- How to prevent duplicates from having inconsistent states?
- According to changes that occurred after the checkpoint, Update node maps
- To recover, read checkpoint and roll forward, i.e., replay any checkpointing sequence
- Two checkpoint regions on disk, swap between them to make
- Checkpoint time (last)
- Point to last segment written (end of the log)
- Pointers to blocks in the index map and segment index table
- Periodically write a checkpoint which contains
and, in particular, to support disconnected operation.

Coda uses caching (replication) to improve availability.

This avoids or minimizes the case where a remote server

Tips for network disconnectedness like NPS and AFS (we use the

Motivation:

Ways to Provide Replication:

- Assign different machines for different partitions
  (tables or rows) which can be updated independently.
- Open partition the database into small regions (e.g., cities),
  e.g., Coda.

Single site machines a must. Secret periodicity sends a

Site machines need to be absolutely identical.

Data replicas can differ apart.

Because

Replicated sites machines: every server receives the same

- Interfaces changed, not the whole
- Copy becomes available
- Database kept in extra copy to be swapped in at the original

Other Replication Techniques:

- Use multiple servers and the most common answer

Coda — Design Overview

Key Principles

- Optimistic replica control
  - Server to handle callbacks
  - NPS’s statesless approach. Although we need some state on the
  - Keep functionality on clients, no servers. Similar in spirit to

If there is a consistency problem

On reconciliation, synchronize state with server, not the user

Typical caching, used entirely on previous access pattern.

Handle critical data to improve disconnected operation (e.g.,

Use callbacks for code coherence

Coda
Coda — Optimistic vs. Pessimistic Replica Control

- **Pessimistic**: Before changing a file, make sure every other client knows that you’re writing it
  - Advantage: prevents conflicts
- **Optimistic**: Let anyone write anytime, and try to resolve conflicts when they occur
  - Advantage: can write even when disconnected

Coda uses optimistic replica control, since allowing disconnected operation is an important goal.

---

Coda — Hoarding

Which files should be hoarded on the local machine?

- Any file currently or recently in use (dynamic priorities)
- The hoard profile contains a list of files to hoard, and their priorities
- When connected, do a hoard walk every 10 minutes to re-establish equilibrium by replacing low-priority files with higher-priority ones from the server

---

Coda — Caching and Callbacks

Coda uses callbacks to maintain cache coherence.

- Client loads whole file into local cache on open (if it’s not there already)
- Client keeps server informed as to which files are in its cache
- Server contract: “I will tell you immediately when your copy of a file is no longer valid, i.e., on callback break”

---

Coda — States

Hoarding ~ Emulation ~ Reintegration

- **Hoarding**: The usual “connected” state. Maintains the cache, trying to keep files in the hoard profile cached
- **Emulating**: The usual “disconnected” state. Serves files from the cache, does security checks locally, and logs changes for reintegration
- **Reintegration**: Synchronizes with the server, bailing out if there are any conflicts.
Know directory file semantics

Handle directory specially — don't treat it like a directory file.

Write log once.

Write out a log file containing the entire transaction and write out a log file containing the entire transaction when the client explicitly asks for it.

If someone else has modified the file (the server has changed)

If a client has changed, then on server, replay changes; unlock changed

What does the client do on recognition?

Coda — Reintegration