Coda

Reference:

- M. Satyanarayanan, *et. al*, Coda: A Highly Available File System for a Distributed Workstation Environment, *IEEE Trans. on Computers*, April 1990.

Coda is a file system for a large-scale distributed consisting of a large collection of untrusted Unix clients and a smaller number of Unix file servers.

Goals:

- constant data availability
- graceful integration of portable computers

Factors that affected the design

- a desire to produce a highly scalable system
- the range of failures addressed
- the need to emulate Unix file semantics

Coda retains the features of AFS that contribute to its scalability and security:

- It uses the model of a few trusted servers and many untrusted clients.
- Clients cache entire files on their local disks.
- Cache cohenrece is maintained by the use of callbacks.
- Clients dynamically map files to servers and cache this information.
- It uses token-based authentication and end-to-end encryption integrated with its communication mechanism.

Failures

- server replication – storing of copies of files at multiple servers, to make shared storage repository highly available. (Fig 1, p. 6 [KiSa92])
- disconnected operation – a client relies solely on cached data.
- optimistic consistency strategy
  1. to provide higher availability (compare to pessimistic strategy)
  2. to support portable workstations
  3. relatively infrequent conflicting updates in academic Unix environments.

Unix Emulation

- Making a failure-proof shared Unix files sytem requires one-copy only Unix semantics.
- Coda semantics, intuitively:
  - A successful file open yields the most recent copy among the set of currently accessible servers.
  - If no server is accessible, the cached copy of the file is used.
  - A successful close indicated that the file has been propagated to the set of currently avaialbe servers.
  - If no server is available, the file is marked for propagration at the earliest opportunity.
- the use of callbacks and an optimistic replication scheme weakens the intuitive semantics. (p. 450)

Disconnected Operations

- Disconnected operation is transparent to a user unless a cache miss occurs.
- Return to normal operation is also transparent, unless a conflict is detected.

Cache Misses

- In normal (connected) operation, a cache miss is tranparent to the user, except performance degration.
- A Coda user specify a prioritized list of files and diretories for retaining in the cache.

Reintegration

- Venus (a cache managment systems of Coda) executes a sequence of update operations os make server's replicas identical to the cached copy.
- Reintegration proceeds top-down, from the root of each cached valume to its leaves.
- Update operations may fail if
  1. no authentication tokens
  2. inconsistency due to update conflicts – put the data in question in a temporary repository on a server and let the user handle it later.
  3. conflict resolution:
  4. atomatic resolution, except for update/update conflict, remove/update conflice, name/name conflict
  5. replair tools to allow users the manually resolve conflicts. This interface differs from that of the normal file operations.