

# Coda: A Highly Available File System for Distributed Workstation Environment

M. Satya

- Coda predecessor: Andrew file system – high availability, mobility, location transparency
- Coda objectives:
  - availability and scalability
  - fault-tolerance, including support for disconnected mode
  - Emulation of UNIX semantics

# Mechanisms

- Replicated servers
  - ‘volumes’ replicated on one or more servers, consistency enforced “lazily”
- Caching
  - of entire files
  - callback registered when file is cached, and is guaranteed to be valid for certain amount of time
  - Otherwise, check on open if cached file is valid
- Optimistic Replication
  - Read-one, write-all (eventually) approach
    - where “all” may be all servers storing volume which can be reached
  - Inconsistencies later detected

# Mechanisms

- Integration
  - Automate – if possible
  - Leave up to user
- Volume versions, update history, periodic updated/keep-alive messages

# Coda semantics

- UNIX semantics – at open get most recent update
  - If cached file, open should check against server version
- AFS2 – reduce traffic to server, most opens are simply validated, instead register a “callback” – ask server to tell you if file changes (within a certain time period)
- Coda:
  - For a file/volume, collection of servers are replicas
  - On open
    - check all replicas for (most recent) file & cache it
    - Or if callback hasn’t been “broken”, use cached file
  - Register callback only with “preferred” server (dynamically changes)
  - Servers check amongst themselves to detect inconsistencies
  - Client can tell them explicitly if one detected
  - On close – write update to all replicas (or keep in cache for later integration)
  - Keep LRU files in cache, or let client specify which ones should be in cache

# Replica Management

- Key: conflict detection
  - Integration may be simple and automated in some cases, or left to user in others
- History – Last Store ID == recent update: who + timestamp
- Version vector == own version value + estimates for each other replica
- Replicas may be in
  - Strong equality
  - Weak equality
  - dominance or submission