The Coda file system

Coda

- The Coda file system is a descendant of AFS.
- Aims to address several requirements that AFS does not meet – the requirement to provide high availability despite disconnected operation.
- The design requirements for Coda were derived from experience with AFS at CMU and elsewhere involving its use in large-scale distributed systems.

Issues With AFS

- Limited form of replication, especially for accessing widely shared files such as system-wide databases.
- The most common difficulties experienced by users of AFS arose from the failure (or scheduled interruption) of servers and network components.
- A mode of computer use was emerging that AFS did not cater for - the mobile use of portable computers.

CONSTANT DATA AVAILABILITY

- Coda aims to meet all three of these requirements under the general heading of **constant data availability**.
  - Provide users with the benefits of a shared file repository
  - Allow them to rely entirely on local resources when the repository is partially or totally inaccessible.
  - Coda retains the original goals of AFS with regard to scalability and the emulation of UNIX file semantics.

CODA Vs AFS

- In AFS,read-write volumes are stored on just one server, the design of Coda relies on the replication of file volumes to achieve a higher throughput of file access operations and a greater degree of fault tolerance.
- Coda relies on an extension of the mechanism used in AFS for caching copies of files at client computers to enable those computers to operate when they are not connected to the network
- CODA follows an optimistic strategy.

The Coda architecture

- **VENUS :** processes at the **client**.
- They play the front end's role of hiding the service implementation from local client processes.
- **Vice :** processes at file **server** computers.(replica managers)
- **Volume Storage Group (VSG) :** The set of servers holding replicas of a file volume.

Coda file access

- A client wishing to open a file in a volume can access some subset of the VSG, known as the available volume storage group (AVSG).
- Coda file access proceeds with cached copies of files being supplied to the client computers by any one of the servers in the current AVSG.
- On close, copies of modified files are broadcast in parallel to all of the servers in the AVSG

Disconnected Operation

- Disconnected operation is said to occur when the AVSG is empty.
- Effective disconnected operation relies on the presence in the client computer's cache of all of the files that are required for the user's work to proceed.
- The copies of files residing on servers are more reliable than those residing in the caches of client computers.

- The copies of files residing in client computer caches are regarded as useful only as long as their currency can be periodically revalidated against the copies residing in servers.
- In the case of disconnected operation, revalidation occurs when disconnected operation ceases and the cached files are reintegrated with those in the servers.

REPLICATION

- Coda’s replication strategy is **optimistic**, it allows modification of files to proceed when the network is partitioned or during disconnected operation.
- It relies on the attachment to each version of a file of a Coda version vector (CVV)

CVV

- A CVV is a vector timestamp with one element for each server in the relevant VSG.
- Each element of the CVV is an estimate of the number of modifications performed on the version of the file that is held at the corresponding server.
- The purpose of the CVVs is to provide sufficient information about the update history of each file replica to enable potential conflicts to be detected and submitted for manual intervention and for stale replicas to be updated automatically.

- If the CVV at one of the sites is greater than or equal to all the corresponding CVVs at the other sites, then there is no conflict. Older replicas include all the updates in a newer replica.
- When neither v1>=v2 t nor v2>=v1 holds for two CVVs – then there is a conflict: each replica reflects at least one update that the other does not reflect.
- Coda does not,resolve conflicts automatically.
- The file is marked as ‘**inoperable**’ and the owner of the file is informed of the conflict.

- When a modified file is closed, each site in the current AVSG is sent an update message by the Venus process at the client, containing the current CVV and the new contents for the file.
- The Vice process at each site checks the CVV and, if it is greater than the one currently held, stores the new contents for the file and returns a positive acknowledgement.
- The Venus process then computes a new CVV with modification counts increased for the servers that responded positively to the update message and distributes the new CVV to the members of the AVSG.

Example: Consider a sequence of modifications to a file F in a volume that is replicated at three servers, S_1, S_2 and S_3. The VSG for F is $\{S_1, S_2, S_3\}$. F is modified at about the same time by two clients, C_1 and C_2. Because of a network fault, C_1 can access only S_1 and S_2 (C_1's AVSG is $\{S_1, S_2\}$) and C_2 can access only S_3 (C_2's AVSG is $\{S_3\}$).

1. Initially, the CVVs for F at all three servers are the same – say, [1,1,1].
2. C_1 runs a process that opens F, modifies it and then closes it. The Venus process at C_2 broadcasts an update message to its AVSG, $\{S_1, S_2\}$, finally resulting in new versions of F and a CVV [2,2,1] at S_1 and S_2 but no change at S_3.
3. Meanwhile, C_2 runs two processes, each of which opens F, modifies it and then closes it. The Venus process at C_2 broadcasts an update message to its AVSG, $\{S_3\}$, after each modification, finally resulting in a new version of F and a CVV [1,1,3] at S_3.
4. At some later time, the network fault is repaired, and C_2 makes a routine check to see whether the inaccessible members of the VSG have become accessible (the process by which such checks are made is described later) and discovers that S_1 and S_2 are now accessible. It modifies its AVSG to $\{S_1, S_2, S_3\}$ for the volume containing F and requests the CVVs for F from all members of the new AVSG. When they arrive, C_2 discovers that S_1 and S_2 each have CVVs [2,2,1] whereas S_3 has [1,1,3]. This represents a *conflict* requiring manual intervention to bring F up-to-date in a manner that minimizes the loss of update information.

On the other hand, consider a similar but simpler scenario that follows the same sequence of events as the one above, but omitting item (3), so that F is not modified by C_2. The CVV at S_3 therefore remains unchanged as [1,1,1], and when the network fault is repaired, C_2 discovers that the CVVs at S_1 and S_2 ([2,2,1]) *dominate* that at S_3. The version of the file at S_1 or S_2 should replace that at S_3.

Normal Operation

- The behaviour of Coda appears similar to that of AFS.
- The advantages deriving from the replication of some or all file volumes on multiple servers are:
  - The files in a replicated volume remain accessible to any client that can access at least one of the replicas.
  - The performance of the system can be improved by sharing some of the load of servicing client requests on a replicated volume between all of the servers that hold replicas.

Disconnected Operation

- A cache miss prevents further progress and the computation is suspended until the connection is resumed or the user aborts the process.
- It is therefore important to load the cache before disconnected operation commences so that cache misses can be avoided.

Summary

When compared with AFS, CODA enhances availability both by,

- Replication of files across servers
- Ability of clients to operate entirely out of their caches.

Both methods depend upon the use of an optimistic strategy for the detection of update conflicts in the presence of network partitions.

Both mechanisms are independent of each other