To appear in *IEEE Workshop on Workstation Operating Systems*
Nov. 1987, Cambridge, MA

Extended Abstract

Coda: A Resilient Distributed File System

M. Satyanarayanan
James J. Kistler
Ellen H. Siegel

Department of Computer Science
Carnegie Mellon University

Distributed file systems have grown in importance in recent years. As our reliance on such systems increases, the problem of availability becomes more acute. Today, a single server crash or network partition can seriously inconvenience many users. Coda is a distributed file system that addresses this problem in its full generality. It is designed to operate in an environment such as the Andrew system at CMU [3, 5, 2], where many hundreds or thousands of workstations span a complex local area network. Coda aspires to provide the highest degree of availability possible in such an environment. An important goal is to provide this functionality without significant loss of performance.

Like Andrew, Coda distinguishes between clients from servers and uses caching of entire files as its remote access mechanism. In addition to improving scalability, whole-file transfer simplifies the handling of failures since a file can never be internally inconsistent. Coda masks server failures and network partitions to the fullest extent possible. Failures during a file operation are totally transparent at the user level unless the operation requires data that is neither cached locally nor present at any accessible server.

Aggregates of files, called Volumes [6], are replicated at multiple server sites. When a file is fetched, the actual data is transferred from only one server. However, the other available servers are queried to verify that the copy of the file being fetched is indeed the most recent. After modification, the file is stored at all the server replication sites that are currently accessible. To achieve good performance, Coda exploits parallelism in network protocols. We have an implementation of a parallel RPC mechanism that is capable of using multicast, if available. This mechanism can transmit files in parallel to multiple sites.

Consistency, availability and performance tend to be mutually contradictory goals in a distributed system. Coda will provide the highest availability at the best performance. A close examination of the way files are shared in an actual file system indicates that an optimistic policy regarding consistency is likely to be successful. Two principles guide the design of consistency mechanisms in Coda. First, the most recently updated copy of a file that is physically accessible must always be used. Second, although inconsistency is tolerable, it must be rare and always detected by the system. We may experiment with heuristics based on file access patterns to resolve simple cases of inconsistency. As in Locus [4], inconsistency is detected by the use of version vectors. However, Coda uses atomic transactions at servers to ensure that the version vector and data of a file are mutually consistent at all times.

At the present time Coda is in the detailed design phase. The implementation of the parallel RPC mechanism has been completed, but the bulk of the design and implementation work remains to be done. This includes areas such as recovery from failures, detection and resolution of inconsistency, file transfer protocols, and support for partitioned operation. The evaluation of Coda along the dimensions of performance and resiliency will also require considerable effort. Although much work remains, we expect that our use of the Andrew file system as a base, Camelot [7] for transaction support, and Mach [1] for operating system support will simplify implementation.

References

[1] Accetta, M., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, A. anc
Mach: A New Kernel Foundation for UNIX Development.
In Proceedings of the Summer Usenix Conference. July, 1986.

[2] Howard, J.H., Kazar, M.L., Menses, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N. and West, M.J.
Scale and Performance in a Distributed File System.
In Proceedings of the 11 th ACM Symposium on Operating System Principles

[3] Morris, J. H., Satyanarayanan, M., Conner, M.H., Howard, J.H., Rosenthal, D.S. and Smith, F.D.
Andrew: A Distributed Personal Computing Environment.
Communications of the ACM 29(3), March, 1986.

[4] Popek, GJ. and Walker, B.J.
The LOCUS Distributed System Architecture.
The MIT Press, 1985.

[S] Satyanarayanan, M., Howard, J.H., Nichols, D.N., Sidebotham, R.N., Spector,
The ITC Distributed File System: Principles and Design.
In Proceedings of the 10th ACM Symposium on Operating System Principles.

[6] Sidebotham, R. N.
Volumes: The Andrew File System Data Structuring Primitive.
In European Unix User Group Conference Proceedings. August, 1986.
Also available as Technical Report CMU-ITCt53, Information Technology Ce
Carnegie Mellon University.

[7] Spector, A.S., Thompson, D., Pausch, R.F., Eppinger, JL., Duchamp, D., Draves, R., Daniels, D.S. and Bloch, JJ.
Camelot: A Distributed Transaction Facility for Mach and the Internet.
Technical Report CMU-CS-87-129, Department of Computer Science,
Carnegie Mellon University, June, 1987.