Efficient Recovery in Harp

Barbara Liskov
Sanjay Ghemawat
Robert Gruber
Paul Johnson
Liuba Shrira

Laboratory for Computer Science
Massachusetts Institute of Technology

1. Introduction

Harp is a replicated Unix file system accessible via the VFS interface. It provides highly available and reliable storage for files and guarantees that file operations are executed atomically in spite of concurrency and failures. Replication enables Harp to safely trade disk accesses for network communication and thus to provide good performance both during normal operation and during recovery. In this position statement, we focus on the techniques Harp uses to achieve efficient recovery.

2. Harp overview

Harp uses the primary copy replication technique [1, 7, 8]. In this method, client calls are directed to a single *primary* server, which communicates with other *backup* servers and informs them about the call. When the backups acknowledge receipt of this information, the operation can commit. At this point the primary returns any results. The second phase, in which the backups are informed about the commit, happens in the background.

When a failure or a recovery from a failure occurs, Harp runs a failover protocol called a *view change* [3, 4, 5]. The result of a view change is a reorganization, in which a failed node is removed from service, or a recovered node is put back into service. The result of such a reorganization is called a *view*; The primary of the new view may be a different node than the primary of the old view.

Harp is one of the first implementations of a primary copy scheme that runs on conventional hardware. It has some novel features that allow it to perform well. Harp achieves good performance by recording the effects of recent modification operations in a log that resides in volatile memory; operations in the log are applied to the file system in the background, while old log entries already applied to the file system are garbage collected. Essentially, this removes disk accesses from the critical path, replacing them with communication (from the primary to the backups), which is substantially faster if the servers are reasonably close together.

In using the log to record recent modifications, Harp is relying on a write-behind strategy, but the strategy is safe because log entries are not lost in failures. We equip each server with a small uninterruptible power supply (UPS) that allows it to run for a short while (e.g., a few minutes) after a power failure; the server uses this time to copy information in the log to disk. The combination of the volatile log and the UPS is one of the novel features of Harp.

Harp provides reliable storage for information. Information survives individual node failures because it exists in volatile memory at several nodes. It survives a power failure because of the UPS's. Also, Harp uses techniques that attempt to preserve information in the face of simultaneous software failures

3. Servers

As with any replication scheme that tolerates network partitions, we require a group of 2n + 1 servers in order to continue to provide service to clients in the presence of n server failures [2]. Unlike the traditional replication methods that keep file copies at all servers, we store only n + 1 copies, since this is enough to allow information to survive n failures. The other n servers need to participate in view changes but need not store copies of the file. We call these additional servers *witnesses* following the terminology of Paris [9], where the idea was first proposed.

In the current Harp implementation, a group has three members and therefore continues to provide service in the presence of any single failure. (Our algorithms also work for groups of other sizes.) Such a replica group has a designated primary, one designated backup, and one designated witness, and in any view there will be a primary and a single backup. A replica group manages the files of one or more Unix file systems, i.e., for a particular file system, all files will have copies at the same group. Only the designated primary and backups store copies (on disk) of the files in the file systems managed by that group.

A good way to organize our system is to arrange for each node to act as the designated primary of one group, the designated backup of another, and the designated witness of a third. This organization allows us to distribute the load among the servers. We call it a "load-sharing" configuration.

4. Failover

We want Harp failovers to be fast in the most common cases of single failure and recovery from a single failure. This is important because client requests are not serviced while a view change is in progress.

Our view change algorithm follows the view change algorithms in [3, 4, 5] and is extended to deal with witnesses. Groupmembers always run within a particular *view*. When a view is formed, it contains at least two group members, one of which acts as primary and the other as backup. If the designated primary is a member of the view, it will act as primary in that view; otherwise the designated backup will be the primary. The designated witness will act as the backup in any view that is missing one of the two others; in this case, we say the witness has been *promoted*. A promoted witness will be *demoted* if a later view change leads to a view that contains the designated primary and backup.

A witness takes part in normal processing of file operations while it is promoted. A promoted witness appears just like a backup as far as the primary of its view can tell, but it differs from a backup in two important ways:

1. Since it has no copy of the file system, it cannot apply its committed operations to the file system.
2. It never discards entries from its log.

When a witness is promoted it receives all log records that are not guaranteed to have reached the disks at both the designated primary and backup. It appends new entries to this log as the view progresses, retaining the entire log until it is demoted. Older parts of its log are stored on a non-volatile device; we use tape drives for witness log storage in our initial system.

There are several points to notice about this approach. First, because of the way the witness' log is initialized when it is promoted, we can bring a node that has not suffered a media failure up to date quickly simply by restoring its log from the witness's log.

Second, every committed operation is recorded at two servers even in a view with a promoted witness. Thus we are providing stable storage [6] for committed operations. Stable storage is most important in a view that lasts a long time, since the probability of a second failure increases with time.

Third, if a view with a promoted witness lasts a very long time, the log may become so large that keeping information in this form is no longer practical. A log can be processed to remove unneeded entries, but even this might not be sufficient to keep the log size practical. In such a case, the best solution may be to reconfigure the system, changing the group membership, and bringing a different node into the group to take over the role of the missing member. This new member would have its file system state initialized by the view change algorithm.

We now consider what happens in a view change and discuss the optimizations that make it run fast. The designated primary and backup monitor other group members to detect changes in communication ability; a change indicates that a view change is needed and will cause a message exchange to reach agreement on the members of the new view. For example, a designated primary or backup might notice that it cannot communicate with its partner in the current view or that it can communicate after not being able to. A view change will be slow if many agreement messages are sent. Therefore, the witness does not monitor the other group members, and it never starts view changes. To avoid the cost of simultaneously initiated view changes (e.g., after recovery from a power failure), we delay the designated backup so that the designated primary will be highly likely to accomplish the view change before the backup can interfere.

A view change will be slow if some group member needs to receive lots of information in order to get up to date. To avoid this potential delay, nodes try to get up to date before a view change. If the designated primary or backup fails, the remaining one will become a primary, and the witness is promoted. The witness is sent the portion of the log containing entries not eligible yet to be garbage collected. There might be many such entries, so we can avoid sending large amounts of information by keeping the witness as a "warm" standby. During normal operation, the primary sends log records to the witness as well as the backup, but the witness does not acknowledge these messages and simply discards records from the bottom of its log when its log exceeds the maximum size. A "standby" witness does not write anything to tape. An interesting question is the effect of the witness's warmup in a load-sharing configuration.

Now suppose the designated primary or backup recovers from a failure. First it will bring itself up to date by communicating with another group member. If there has just been one failure, this group member will be accessible and a member of an active view. If the recovering node has not had a media failure, it will communicate with the witness; this is likely to have little, if any, impact on the speed of processing operations.

If the recovering node has had a media failure, it must communicate with the primary of the current view, and then with the witness. We have designed an algorithm that allows the primary to continue processing operations while it sending a consistant file system state. Thus users continue to receive service, but probably with a somewhat degraded response time.

After the recovering node is up to date, it does the view change, but the view change should be short, since little information is exchanged.

5. Conclusion and Status

In this position paper we have described how Harp achieves efficient failover in the most common cases of failure and recovery. This is important because client requests are not serviced while a wiew change is in progress. Harp shortens view changes by introducing a warmup phase that allows servers to get up to date without service interuption. The warmup phase also simplifies the view change algorithm because it needs to be optimized for fewer cases.

The normal case Harp algorithms are operational. We are completing now the view change part. The backgroung warmup of the recovering primary and backup is implemented, while the warmup of witness is not in place yet. Once the view change is operational, we will measure and tune its peformance.

References

s
}

1. Alsberg, P., and Day, J. A Principle for Resilient Sharing of Distributed Resources. Proc. of the 2nd International Conference on Software Engineering, October, 1976, pp. 627-644. Also available in unpublished form as CAC Document number 202 Center for Advanced Computation University of Illinois, Urbana-Champaign, Illinois 61801 by Alsberg, Benford, Day, and Grapa..

2. Bernstein, P. A., Hadzilacos, V., and Goodman, N.. *Concurrency Control and Recovery in Database Systems.* Addison Wesley, 1987.

3. El-Abbadi, A., and Toueg, S. Maintaining Availability in Partitioned Replicated Databases. Proc. of the Fifth Symposium on Principles of Database Systems, ACM, 1986, pp. 240-251.

4. El-Abbadi, A., Skeen, D., and Cristian, F. An Efficient Fault-tolerant Protocol for Replicated Data Management. Proc. of the Fourth Symposium on Principles of Database Systems, ACM, 1985, pp. 215-229.

5. Gifford, D.K. Information Storage in a Decentralized Computer System. Technical Report CSL-81-8, Xerox Corporation, March, 1983.

6. Lampson, B. W., and Sturgis, H. E. Crash Recovery in a Distributed Data Storage System. Xerox Research Center, Palo Alto, CA, 1979.

7. Oki, B. M., and Liskov, B. Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems. Proc. of the 7th ACM Symposium on Principles of Distributed Computing, ACM, August, 1988.

8. Oki, B. M. Viewstamped Replication for Highly Available Distributed Systems. Technical Report MIT/LCS/TR-423, MIT Laboratory for Computer Science, Cambridge, MA, 1988.

9. Paris, J-F. Voting With Witnesses: A Consistency Scheme for Replicated Files. Proc. of the 6th International Conference on Distributed Computer Systems, IEEE, 1986, pp. 606-612.

Table of Contents