A Hybrid Model for Mobile File Systems

John Saldanha and David L. Cohn
Distributed Computing Research Laboratory
University of Notre Dame, Notre Dame, IN 46556
{jes,dlc}@cse.nd.edu

Abstract

Existing distributed file systems are based on either a client-server model or a peer-to-peer model. We believe that the dynamic conditions of mobile computing and new classes of devices such as PDAs will no longer permit rigid adherence to either of these models. In this paper, we argue that a hybrid of the two will have to be used. We then propose a file system design based on such a model which exploits the ability of highly portable devices like PDAs to be carried by their owners at all times. We examine the relationship between mobile computing personae and file systems and show how the persona concept can be used to support the file system needs of mobile users. Finally we discuss some issues raised by the implementation we are building.

1. The Need for a Hybrid Model

In the client-server model, used by distributed file systems such as AFS [3] and Coda [4], a select subset of the machines are given the special status of *servers* and act as the primary storage sites for files. At the remaining machines (the *clients*), files are cached locally for reasons of performance and availability. However, these cached replicas may not be accessed from anywhere else in the system. They are therefore "second-class," compared to the replicas on the servers, which are "first-class." By contrast, in the peer-to-peer model, used by file systems such as Ficus [2], all machines have equal status and so all replicas are first-class.

A distributed file system that uses the client-server model provides several things users want:

- Security - access to files is under the control of their owners.
- Integrity - files are not lost due to crashes, disk failures, etc.
- Location transparency - the user sees the same name space at all locations.
- Capacity - large disk capacity is available at a reasonable price.
- Accessibility - files can be shared with other users in a controlled manner.

The use of trusted and reliable servers is critical to providing each of the above attributes. However, exclusive reliance on servers results in limited availability. A solution to this problem is to use the servers of a distributed file system as the "permanent home" for files but permit the user to temporarily store and access these files elsewhere.

Coda took an important step in this direction through its support for disconnected operation. However, the Coda solution only works if *all* the computers operated by a user are clients of the same distributed file system. There are many situations in which this does not hold. For example, while at home, the user may work on a stationary workstation that has no network connectivity. Coda files cannot be accessed on this workstation even if it can communicate over a short-range link with a portable Coda client carried by the user. Another problem is that Coda does not allow sharing of files between clients which are connected to each other but that are disconnected from all servers.

Coda maintains first-class replicas only at servers and allows second-class replicas only at clients. We propose to extend this model and allow second-class replicas even at machines which are not clients if they are occasionally able to communicate with a client. Coda does not allow direct sharing between second-class replication sites. We suggest that such sharing be allowed when no first-class replicas are accessible. Allowing such sharing introduces a coherency problem between the second-class replicas. This is an interesting and significant problem which needs to be addressed. Figure 1 graphically depicts the replication model being proposed. The shaded element represents a replica that previously could not exist and the grey

0-8186-6345-6/95 $04.00 © 1995 IEEE

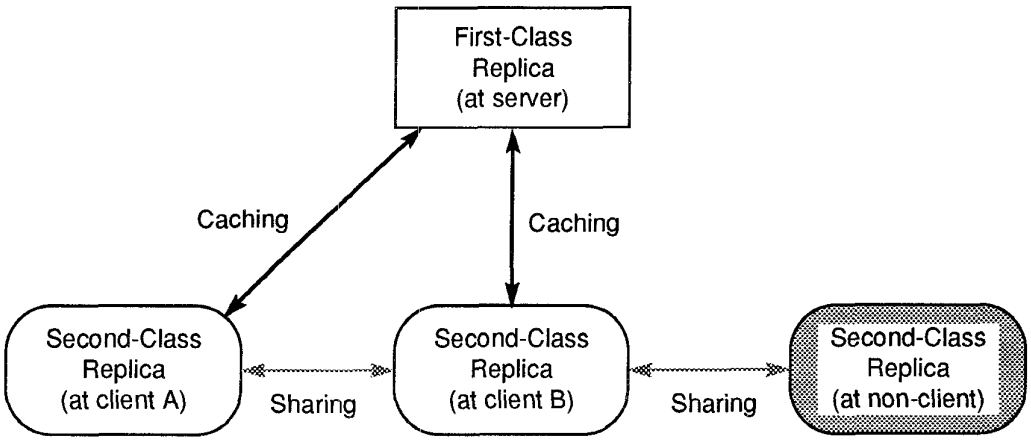

Figure 1 Proposed replication model

arrows depict sharing interactions that previously could not occur.

2. A PDA-based File System Design

Most mobile computers currently in use are *portable workstations*. That is, they are lighter and smaller than desktop workstations but are functionally very similar to them. We use the term *PDA* to refer to machines that sacrifice some of this functionality for improved portability and extended battery life.

Existing file system support for mobile computing, which has been designed for portable workstations, may not work well for PDAs. For example, a typical scenario in which Coda works effectively is as follows: the user brings his or her portable workstation into the office, plugs it into the network, works on it for a while, unplugs it when it is time to leave and then uses it in disconnected mode. Because the user was working on the portable just before disconnection, the active files are automatically loaded into the cache.

Unfortunately, this scenario is unlikely if the portable is a PDA. Because of the PDA's tiny user interface, the tasks for which it is well suited are limited. Therefore, whenever a user has access to a larger computer, he or she is likely to use that one instead of the PDA. Thus, a much more likely scenario is that while the user is in the office he or she will use a desktop system. Then, with standard Coda, when he or she leaves the office, the active files would not be in the PDA's cache; they would be in the desktop's cache.

In fact, because of its limited user interface, a PDA should probably be used in *conjunction* with larger computers when they are accessible. Let us consider a hypothetical example. A user preparing for a business trip starts working on a document using an office desktop system. Through a new file sharing mechanism, a copy of the document is concurrently maintained in the cache on a PDA. Later, on the cab ride to the airport, the user would employ the PDA to edit the document. Once on the airplane however, the user might find that the seatback in front has a computer that can be used to edit the document. What is needed to make all this work is a mechanism for transparently sharing files between a PDA and a proximate computer.

While the PDA's small size is a limitation, it also brings with it the tremendous advantage that it can be carried by its owner at all times. This feature is something we must exploit in designing a file system for a mobile computing environment that includes these devices. Since the PDA is with its owner at all times, it can act as a carrier of currently active or frequently used files.

One solution to the file system needs of a mobile user is a PDA-based file system. In such a file system, special status is accorded to the user's PDA because it is always available. The permanent home of the files would still be on servers, and the PDA would be a link between the distributed file system and the file systems of other, independent computers.

Whenever circumstances permit, the user works at a connected client of the distributed file system. At such times, the PDA, through a connection with this client or through a direct connection with a server, hoards important files (specified by a hoard file) and currently active files. This situation is shown in Figure 2a. When the user has to disconnect from the distributed file system, he or she then depends on the contents of the PDA's cache.

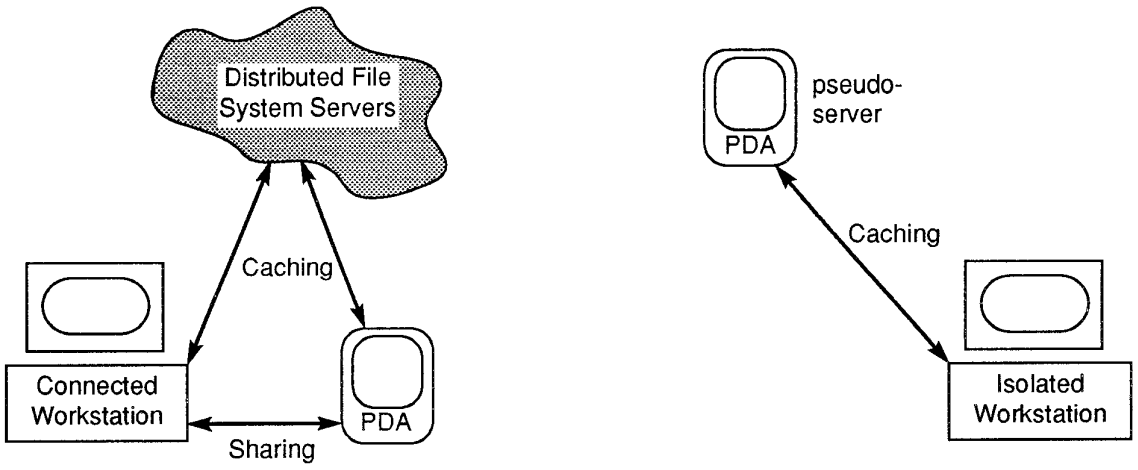

Figure 2 A PDA-based file system

While disconnected, files can be directly accessed on the PDA. However, as shown in Figure 2b, the PDA can also act as a pseudo-server to an isolated workstation (one that is not a client of the distributed file system). This workstation caches files that the user needs. When a file is modified at this workstation, or when a new file is created there, a copy is sent to the PDA. When the user next connects to the distributed file system, the PDA propagates all changes made while disconnected back to the real server(s).

An argument against entrusting a PDA with the responsibility of carrying files critical to its owner is that a PDA is vulnerable to loss, theft or damage. However, we are relying on the PDA only while the user is disconnected from the distributed file system. Therefore, only files that have been created or modified since the last disconnection are vulnerable to loss. This is a reasonable risk in exchange for increased availability. Also, copies (possibly stale) of lost files may exist on other workstations, providing some additional protection.

3. Computing Personae and File Systems

A user's computing persona [1] is, roughly speaking, the computing environment that he or she sees. This includes a name space, which is generally a name space of files, but which could also be extended to include other objects such as printers. It also includes the user's open files, available devices, user interface, communication connections, active applications and the applications' states. Some of these components, such as files and network connections, are passive, while others, including running applications and communication connections, are active.

Today, a user with access to multiple computers has a computing persona on each one. He or she has to manually force these personae to resemble each other by copying files, restarting applications, etc. Ideally, the user should be able to move from one location to another and have the persona follow.

An active entity called a *persona manager*, resident at one or more locations in the system, could monitor the persona and facilitate its movement. When the user logs out at one location, the persona manager "freezes" the persona, which then remains dormant until the user logs in again (at a possibly different location). At login, the persona manager "unfreezes" the persona and re-establishes it at the user's (new) location.

The relationship between the file system and the persona manager is two-way. On the one hand, the user's *view* of the file system is part of the persona, and so it is the persona manager's job to manage and move it when needed. On the other hand, the persona manager needs support from the file system to perform its functions. For example, in order to restart the user's applications at a new location from the same point at which they were left off, the persona manager would store the state of these applications somewhere in the file system.

In a persona-based file system, when the user moves from one location to another, any files created or modified at the old location move along with the user to the new location. If both locations are part of a connected system,

this can be handled automatically by a distributed file system. Otherwise, the persona manager will have to perform this transfer and is entrusted with the responsibility for all modified and newly created files. However, as soon as the user moves to a connected client of a distributed file system, these modifications are written to a server, and the persona manager is relieved of its burden.

It should be noted that a persona-based file system could also be PDA-based. There is no magical way of moving a user's persona from one location to the other. There are only two ways for this transfer to occur. If there is a connection (wired, wireless or a combination of the two) between the old and new location, the transfer can use this connection. If not, the only alternative is for the user to physically carry the persona on some device. A PDA is the most logical choice for such a device because, as we assume, it will always be with the user.

4. Implementation

We are currently building a Coda-based implementation to evaluate the feasibility and usefulness of the ideas we have presented. We are modifying the Coda client to enable it to service requests from a system that is not a Coda client, and are developing a vnode-based virtual file system to make Coda files available on such non-clients. We do not foresee major changes to the Coda client to enable it to act as a pseudo-server since it is already built to emulate the server during periods of disconnection. However, the changes required to allow sharing of files between Coda clients should prove more tricky since this has the potential for introducing incoherency between replicas. We propose to address the coherency issue by associating a write-token with each file. While replicas can exist at multiple hosts, the write-token always moves with the computing persona of the file's owner, thus preventing any write-write conflicts.

5. Conclusion

We have argued that the increasing mobility of users and devices makes it necessary for distributed file systems to use a mix of the client-server and peer-to-peer models. We have presented a design for a distributed file system that exploits the ability of highly portable devices like PDAs to be carried by their owners at all times. We have also shown how the concept of mobile computing personae may be applied to addressing the file system needs of mobile users. We are building an implementation to test our ideas and have discussed some of the issues raised by this exercise.

6. References

[1] A. Banerji et al., "Mobile Computing Personae," **Fourth IEEE Workshop on Workstation Operating Systems**, Oct. 1993, Napa, CA, pp. 14-20.

[2] J. Heidemann et al., "Primarily Disconnected Operation: Experiences with Ficus," **Second Workshop on Management of Replicated Data**, Nov. 1992.

[3] J. Howard et al., "Scale and Performance in a Distributed File System," **ACM Trans. on Computer Systems 6(1)**, Feb. 1988, pp. 51-81.

[4] J. J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System," **ACM Trans. on Computer Systems 10(1)**, Feb. 1992, pp. 3-25.