AFS and the Web: Competitors or Collaborators?

M. Satyanarayanan
Carnegie Mellon University
Pittsburgh, PA

Mirjana Spasojevic
Hewlett-Packard Laboratories
Palo Alto, CA

July 1996
CMU-CS-96-153

To appear in
Proceedings of the Seventh ACM SIGOPS European Workshop,
Connemara, Ireland, September 1996

Abstract

The World-Wide Web and AFS represent two different approaches to the problem of large-scale information sharing. The goal of this paper is to critically compare these two mechanisms and to expose their relative strengths and weaknesses. Our comparison shows that the Web and AFS are not really competing technologies. Rather, they represent complementary technologies that may be used together for mutual advantage. We present real-life examples to confirm that this potential can indeed be realized in practice.

This research was supported by the Air Force Materiel Command (AFMC) and the Advanced Research Projects Agency (ARPA) under contract number F19682-93-C-0193.The U.S. government is authorized to reproduce and distribute reprints for government purposes, notwithstanding any copyright notation thereon.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the AFMC, ARPA, HP, CMU, or the U.S. Government.

Keywords: World Wide Web, Mosaic, Netscape, Explorer, AFS, Andrew, scalability, performance, API, GUI, Unix, caching, security, replication, location transparency, heterogeneity, NCSA, Transarc, CMU, information repositories, world-wide information systems, distributed systems

1. Introduction

Over the last few years, the World-Wide Web [2] has risen to dominance as a mechanism for wide-area information access. Each day brings new reports of the growth of the Web, and this trend shows no signs of abating any time soon. To many people, the Web and the Internet are synonymous. Unfortunately, success has exposed many limitations of the Web such as its tendency to overload the network and servers, its limited ability to control access to sensitive data, its lack of mechanisms for data consistency, and its susceptibility to network and server failures. It is now widely recognized that these problems must be solved for the continued growth of the Web.

With much less fanfare, another world-wide information system, AFS, has also been operational on the Internet. AFS was originally designed to support the file sharing needs of a campus-sized community of five to seven thousand workstations [11, 15]. In current parlance, AFS was conceived as an information sharing mechanism for the *Intranet* of an organization. Since then, AFS has been evolved to function effectively over the Internet [17, 21]. Many organizations have been part of a single distributed Unix file name space supported by AFS. As of 1994, this system spanned well over 100 organizations world-wide, with each organization typically containing many tens or hundreds of clients. Measurements reported in a recent paper [20] confirm that AFS does indeed function effectively at this scale.

The goal of this paper is to critically compare these two mechanisms for world-wide information access. We begin by asking the following questions:

- What are the relative strengths and weaknesses of the two mechanisms?
- Which of these differences are superficial, and which are deep?
- Why is the Web so much more visible and popular?

In performing this comparison, it is important to keep in mind that the two mechanisms are not addressing precisely the same problem. While AFS has the relatively narrow and well-defined goal of provided distributed Unix file access, the goal of the Web is broader and less explicit. Further, AFS is now stable and mature while many aspects of the Web are still evolving. In spite of these caveats, we believe that a comparison of the two mechanisms will provide useful insights.

Our comparison shows that the Web and AFS are not really competing technologies. Rather, they represent complementary technologies that may be used together for mutual advantage. We present real-life examples to confirm that this potential can indeed be realized in practice.

2. Comparison of Characteristics and Evolution

2.1. Interface Level and Usability

AFS supports an application programming interface (API) and is primarily intended for use by programs. Consistent with the Unix heritage of AFS, the user interface is only of secondary importance. In contrast, the Web interface is primarily a graphical user interface (GUI) for humans. While one can write code to parse and interpret Web pages, this is not the intended use of the Web and is unlikely to be efficient.

The ease of use of the Web via browsers like Mosaic and Netscape Navigator has made it appealing even to novice users. The visual impact of tastefully mixed text, graphics and animation on potential new users is enormous. These factors have played a key role in the expansion of the Web user community. In contrast, AFS' strengths are harder to explain to an unsophisticated audience. By its very nature, an API is harder to demonstrate than a GUI. Only the relatively small Unix community appreciates and values the primary strength of AFS — its ability to preserve the Unix file system API at large scale. Since Unix is unfriendly to novice users, the rate of growth of this community is low.

2.2. Targeted Access versus Browsing

AFS assumes that the pathnames of objects to be accessed are known with accuracy. The only aids to search are the mnemonic significance of pathnames and the directory organization. Consistent with the Unix file system model, AFS directories offer no auxiliary annotation to help in browsing. While AFS' location transparency hides server names, its hierarchical name space inevitably reflects some aspects of administrative and organizational structure.

In contrast, a Web page is merely assumed to be the starting point of an exploration. The ability to mix annotational information with pointers to other Web pages greatly simplifies *browsing.* These pointers are often to pages in different administrative and organizational domains. Thus, a Web URL can be viewed as a *fuzzy pointer,* whose dereferencing can result in widely varying targets depending on context. This builtin ability to cope with imprecision is of enormous value in wide-area information access, because it is a good match for human cognitive limitations.

2.3. Workload Characteristics

The different usage models of AFS and the Web result in substantial differences in client and server workload characteristics in the two systems. AFS references exhibit substantial temporal locality, thus making client caching useful. In contrast, Web references from a client tend to exhibit poor temporal locality; this renders conventional client caching much less effective [1, 3, 5].

On the other hand, there is substantial *collective* locality in the Web references of an organization. In other words, Web documents that are accessed by one user in an organization are likely to also be accessed by other users in that organization. This suggests the utility of a shared organizational cache [10]. A variant of this idea uses geography rather than organizational boundaries as the basis of collective caching [6]. In contrast, intermediate caches have been shown to be of little value in AFS [12].

There are also important differences with respect to spatial locality. Once an AFS file is opened, it is likely to be read sequentially in its entirety. This makes simple sequential read-ahead useful. Web traffic additionally exhibits a more complex form of spatial locality [1, 5] — chains of URL references that tend to occur in the same sequence each time. Exploiting this form of spatial locality requires a prefetching mechanism that is considerably more sophisticated than simple read-ahead.

In addition, AFS workloads are influenced by the fact that servers maintain state information regarding clients. In contrast, Web servers are stateless and maintain no information about clients. Although maintaining state about clients at servers increases the complexity of a distributed system, it has been shown to play a critical role in improving performance and ensuring correctness [13].

2.4. Consistency and Replication

Since AFS allows remote objects to be updated by clients, its workloads include both read and write traffic. The cache coherence protocol guarantees that when a client closes a file, any updates it has made are visible on subsequent opens at all other clients. In contrast, the Web does not allow pages to be updated by clients, nor does it provide any guarantees regarding the visibility of updates at servers. From the viewpoint of clients, the Web is effectively composed of *read-only* objects, with lazy propagation of updates from servers to clients. The use of *forms* allows clients to supply new data, but the modicum of updateability provided by this mechanism hardly compares with full-fledged support for updates.

The location transparency offered by AFS simplifies replication. Server names are never exposed to applications or users; instead, the name resolution mechanism of AFS transparently maps pathnames to servers. The hooks for substitution of one replica for another are thus naturally present in the design of AFS. As a consequence, read-only replication at the granularity of entire volumes was supported by AFS early in its evolution. Further, the feasibility of read-write replication has been demonstrated by Coda [16, 17], a descendant of an early version of AFS.

In contrast, Web URLs explicitly identify server names. Hence, the use of replication for availability or load-balancing, implies one of two consequences: exposure of replication to applications, or support for replica transparency at the network protocol level. There is considerable interest in developing a location transparent Web naming mechanism based on the notion of *Uniform Resource Names*, though there is no consensus yet on the details of this mechanism.

2.5. Security

Careful attention has been paid to the security aspects of AFS since its earliest days [14]. Throughout its evolution, AFS has preserved a model of limited trust: a small collection of trusted servers provides a secure storage repository for users at a large number of clients. There is support for secure communication at the RPC level, a distributed authentication service, and a file protection scheme that combines access lists with UNIX mode bits. The security model does not assume integrity of the clients or the network.

In contrast, security has been an afterthought in the evolution of the Web. The early document protection scheme was based on a simplified access control mechanisms that included or excluded IP addresses of clients. With the growth of the Web and its increasing potential for electronic commerce, it has become clear that secure data transfer protocols, authentication and access control must be introduced.

2.6. Scalability

Concern for scalability pervaded the design of the first prototype of AFS [15], and has continued to be the dominant theme of its evolution. Many aspects of AFS contribute to its scalability:

- the use of callback-based caching [7] to minimize server and network load.
- the use of *volumes* for ease of system administration [18].
- attention to security issues, as elaborated in the previous section.
- the decomposition of the system into independent *cells* to support organizational autonomy [17].

In contrast, scalability was not a significant influence on the early evolution of the Web. Only now, in light of the Web's growth, has scalability become an important consideration.

2.7. Heterogeneity

AFS is closely tied to the Unix file system model. Only late in its evolution were efforts made to provide cross-platform support. In contrast, the Web is platform-neutral and browsers became available for most platforms early in its evolution. The ability to access the Web from virtually any client played a significant role in its rise to prominence.

A specific aspect of heterogeneity, namely the ability to execute platform-specific code using a fixed pathname, received early attention in AFS. To support this, AFS translates the pathname component "`@sys`" into a platform-specific string. Though modest in scope and functionality, this support for heterogeneity has proved to be a widely-used feature of AFS. Recent developments allow the Web to offer more comprehensive support for heterogeneity. The advent of Java carries the idea of platform-independence much further, enabling the creation and dynamic execution of code that is platform-neutral.

2.8. Marketing strategy

Web software was distributed free of cost in source code form initially. This enabled potential users at many organizations to experiment with the Web, to appreciate its benefits, and to contribute ideas and code improvements to it. This period of open and free access led to the rapid growth of the Web, effectively creating a market for Web products. Only after the Web was already in widespread use did proprietary browsers such as Netscape Navigator and Microsoft Explorer appear.

In contrast, AFS was never freely distributed either in source or binary form. Although it was created at a university, the terms of the contract under which it was built vested ownership with IBM. As a consequence, AFS could not be freely distributed — each distribution involved explicit licensing negotiations. This substantially increased the entry cost for an organization to use AFS, thereby reducing its popularity. In spite of owning AFS, IBM never made it into a product. Only after Transarc Corporation was formed did AFS became commercially available. But even Transarc views AFS primarily as a stopgap for its successor, DFS [9]. In summary, three distinct factors have hurt the creation of a market for AFS: its lack of free availability early in its evolution, the delay in making it a product, and the perception that it will soon be replaced.

3. Symbiosis

The complementary design characteristics of the Web and AFS suggest the possibility of combining their strengths in a composite system. We describe several such systems in this section.

3.1. NCSA: Server Load Balancing

A recent paper reports on the use of AFS as the shared back end for a scalable Web server at the National Center for Supercomputing Applications (NCSA) [8]. In this design, AFS enables a collection of physical Web servers to masquerade as a single, large, virtual Web server. Load-balancing across physical servers is achieved by a customized Domain Name Service (DNS) resolver that maps the hostname of the virtual Web server to a different physical server in round-robin order. The capacity of the virtual Web server is easily increased by adding more physical servers.

With individual Web servers being AFS clients, the worst-case document access time is increased because every document is moved twice: from an AFS server to the Web server, and then to the Web client. However, the effectiveness of AFS caching ensures that the worst case only occurs rarely. The paper reports that this approach has substantially improved performance and availability at NCSA.

3.2. Carnegie Mellon: Improved System Administration and Security

Another organization that uses AFS to store Web data is Carnegie Mellon University. In this deployment, a small number of Web servers provide access to Web data stored in AFS. Users create and update Web documents from any AFS client. Since AFS servers and clients do not run Web servers, their identity is not part of URLs. This simplifies the addition or deletion of AFS clients and servers because no published or embedded URLs have to be invalidated.

The setup at Carnegie Mellon pays additional attention to issues of security. Web servers are authenticated to AFS as special users, and can only access files in those directories with appropriate ACLs. This gives users fine-grain control over which AFS documents are visible via the Web. The default protection is such that Web access is prohibited.

3.3. Transarc: Reduced Network and Server Load

Like Carnegie Mellon, Transarc Corporation uses AFS as a Web document repository. At Transarc, Web clients have been modified to generate direct AFS references when they encounter URLs referring to documents in AFS [19]. For example, the URL

`http://www.transarc.com/afs/transarc.com/public/www/index.html`

is translated by a Web client into the URL

`file:/afs/transarc.com/public/www/index.html`.

Many benefits follow from this approach: documents are automatically cached, they can be easily replicated and AFS' security model is directly exploited.

4. Conclusion

In summary, AFS and the Web occupy intersecting, but substantially non-overlapping portions of the design space for wide-area information repositories. Many of the ideas for scalability first developed and demonstrated in the context of AFS must be retrofitted into the Web. Conversely, a good way to improve the usability and ease of browsing of AFS is to embed interlinked Web pages in it. A promising approach to combining the strengths of the two mechanisms is for an organization to use AFS in its Intranet, but to provide Internet Web access points into its AFS name space. Thus, both AFS and the Web are important building blocks — neither is likely to displace the other. Approaches that improve the integration of these two mechanisms are more likely to succeed than approaches that try to eliminate either.

While AFS and the Web jointly address many of the basic problems of world-wide information access, they generate new problems that demand our attention. First, the introduction of Java technology in the Web blurs the distinction between passive access to data and active computation on that data. We do not yet understand the full implications of this change on the scalability, security, and manageability of shared repositories. Second, AFS and the Web focus on simplifying access to data. As the space of accessible data grows, searching it to locate relevant information becomes an increasing problem. Indexing facilities such as Lycos, Yahoo and AltaVista, new client abstractions such as dynamic sets [22], and mechanisms for resource discovery such as Harvest [4] represent initial steps toward solving this problem. But the full scope of this challenge is likely to occupy us well into the 21st century.

Acknowledgements

We would like to thank Bob Baron, Mark Crovella, Maria Ebling, David Steere and the anonymous referees for their suggestions in improving the content and presentation of this paper. M. Satyanarayanan was supported in the writing of this paper by the Air Force Materiel Command (AFMC) and ARPA under contract number F196828-93-C-0193. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFMC, ARPA, CMU, HP, or the U.S. Government.

References

[1] Almeida, V., Bestavros, A., Crovella, M., de Oliveira, A.
Characterizing Reference Locality in the WWW.
Technical Report TR-96-11, Boston University Computer Science Department, 1996.

[2] Berners-Lee, T., Calliau, R., Luotonen, A., Frystyk Nielsen, H., Secret, A.
The World-Wide Web.
Communications of the ACM 37(8), August, 1994.

[3] Bestavros, A., Carter, R.L., Crovella, M.E., Cunha, C.R., Heddaya, A., Mirdad, S.A.
Application-Level Document Caching in the Internet.
In *Proceedings of the Second International Workshop on Services in Distributed and Networked Environments (SDNE'95)*. June, 1995.

[4] Bowman, M., Danzig, P.B., Hardy, D.R., Manber, U., Schwartz M.F.
The Harvest Information Discovery and Access System.
Computer Networks and ISDN Systems 28:119-125, August, 1995.

[5] Dharap, C., Bowman, M.
Preliminary Analysis of Wide-Area Access Traces.
Technical Report CSE-95-030, Department of Computer Science, Penn. State Univ., 1995.

[6] Gwertzman, J.S., Seltzer, M.
The Case for Geographical Push-Caching.
In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*. May, 1995.

[7] Howard, J.H., Kazar, M.L., Menees, S.G., Nichols, D.A., Satyanarayanan, M., Sidebotham, R.N., West, M.J.
Scale and Performance in a Distributed File System.
ACM Transactions on Computer Systems 6(1), February, 1988.

[8] Katz, E.D., Butler, M., McGrath, R.
A Scalable HTTP Server: The NCSA Prototype.
Computer Networks and ISDN Systems 27, September, 1994.

[9] Kazar, M.L., Leverett, B.W., Anderson, O.T., Apostolides, V., Bottos, B.A., Chutani, S., Everhart, C.F., Mason, W.A., Tu, S., Zayas, E.R.
DEcorum File System Architectural Overview.
In *Proceedings of the USENIX Summer Conference*. June, 1990.

[10] Luotonen, A., Altis, K.
World-Wide Web Proxies.
Computer Networks and ISDN Systems 27, September, 1994.

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

[12] Muntz, D., Honeyman, P.
Multi-level Caching in Distributed File Systems.
In *Proceedings of the Usenix Winter 1992 Technical Conference*. January, 1992.

[13] Ousterhout, J.K.
The Role of State in Distributed Systems.
CMU Computer Science: A 25th Anniversary Perspective.
In Rashid, R.F.,
Addison-Wesley, 1991.

[14] Satyanarayanan, M.
Integrating Security in a Large Distributed System.
ACM Transactions on Computer Systems 7(3), August, 1989.

[15] Satyanarayanan, M., Howard, J.H., Nichols, D.N., Sidebotham, R.N., Spector, A.Z., West, M.J.
The ITC Distributed File System: Principles and Design.
In *Proceedings of the 10th ACM Symposium on Operating System Principles*. December, 1985.

[16] Satyanarayanan, M., Kistler, J.J., Kumar, P., Okasaki, M.E., Siegel, E.H., Steere, D.C.
Coda: A Highly Available File System for a Distributed Workstation Environment.
IEEE Transactions on Computers 39(4), April, 1990.

[17] Satyanarayanan, M.
Scalable, Secure, and Highly Available Distributed File Access.
IEEE Computer 23(5), May, 1990.

[18] Sidebotham, R.N.
Volumes: The Andrew File System Data Structuring Primitive.
In *European Unix User Group Conference Proceedings*. August, 1986.

[19] Spasojevic, M., Bowman, M., Spector, A.
Using a Wide-Area File System Within the World-Wide Web.
In *Proceedings of the Second International WWW Conference*. October, 1994.

[20] Spasojevic, M., Satyanarayanan, M.
An Empirical Study of a Wide-Area Distributed File System.
ACM Transactions on Computer Systems 14(2), May, 1996.

[21] Spector, A.Z., Kazar, M.L.
Wide Area File Service and the AFS Experimental System.
Unix Review 7(3), March, 1989.

[22] Steere, D.C., Satyanarayanan, M.
Using Dynamic Sets to Overcome High I/O Latencies During Search.
In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*. Orcas Island, WA, May, 1995.