A Usage Profile and Evaluation of a Wide-Area Distributed File System *

Mirjana Spasojevic
Transarc Corporation

M. Satyanarayanan
Carnegie Mellon University

Abstract

The evolution of the Andrew File System (AFS) into a wide-area distributed file system has encouraged collaboration and information dissemination on a much broader scale than ever before. In this paper, we examine AFS as a provider of wide-area file services to over 80 organizations around the world. We discuss usage characteristics of AFS derived from empirical measurements of the system, and from user responses to a questionnaire. Our observations indicate that AFS provides robust and efficient data access in its current configuration, thus confirming its viability as a design point for wide-area distributed file systems.

1. Introduction

Over the last decade, distributed file systems such as AFS and NFS in the Unix world, and Netware and LanManager in the MS-DOS world have risen to prominence. Today, virtually every organization with a large collection of personal machines uses such a system. The stunning success of the distributed file system paradigm is attributable to three factors.

First, a distributed file system simplifies the *separation of administrative concerns* from usage concerns. Users work on tasks directly relevant to them on their personal machines. Incidental but essential tasks such as backup, disaster recovery, and expansion of disk capacity are handled by a professional staff who focus primarily on the servers.

Second, the use of a distributed file system simplifies the *sharing of data* within a user community. Such sharing can arise in two forms: by a user accessing his files from different machines, and by one user accessing the files of another user. The ability to easily access one's files from any machine enhances a user's mobility within his organization. Although the accessing of someone else's files is not a frequent event (a fact confirmed by many previous studies [1, 6]), ease of access once the need arises is perceived as a major benefit by users. In other words, while sharing may be rare, the payoff of being able to share easily is very high[1].

Third, *transparency* is preserved from the users' and applications' points of view. Applications do not have to be modified to use a distributed file system. Because a distributed file system looks just like a local file system, a user does not have to learn a completely new set of commands or new methods of file usage.

The designs of modern distributed file systems reflect these observations. They use a client-server model, offer location transparency, rely on caching to exploit locality, provide fairly weak consistency semantics

*This research was funded by the Advanced Research Project Agency, under contract number MDA972-90-C-0036, ARPA order number 7312. The views and conclusions expressed in this paper are those of the authors and do not represent the official position of ARPA, Transarc Corporation or Carnegie Mellon University.
Please direct correspondence to Mirjana Spasojevic, Transarc Corporation, The Gulf Tower, 707 Grant Street, Pittsburgh, PA 15219.

[1] In this respect a distributed file system is like a telephone system: although a given individual only tends to call a tiny fraction of all telephone numbers, the latent ability to effortlessly reach any other telephone in the world is viewed as a major asset of the system.

relative to databases, and support programming and user interfaces that are close to those of a local file system. The success and widespread usage of these systems confirms the appropriateness of these design choices.

But this success engenders a new question: "Is the distributed file system paradigm sustainable at very large scale?" In other words, how well can a very large distributed file system meet the goals of simplifying system administration, supporting effective sharing of data, and preserving transparency? Growth brings many problems with it [12]: the level of trust between users is lowered; failures tend to be more frequent; administrative coordination is more difficult; performance is degraded. Overall, mechanisms that work well at small scale tend to function less effectively as a system grows. Given these concerns, how large can a distributed file system get before it proves too unwieldy to be effective?

In this paper, we seek to answer this question by studying the usage characteristics of AFS, the largest currently deployed instance of a distributed file system. At the time of writing, AFS unites about 1,000 servers and 20,000 clients in 7 countries into a single file name space. We estimate that more than 100,000 users use this system worldwide. In geographic span as well as in number of users and machines, AFS is the largest distributed file system that has ever been built and put to serious use.

Our study confirms that the distributed file system paradigm is indeed being effectively supported at the current scale of AFS. Further, our data does not expose any obvious impediments to further growth of the system. While asymptotic limits to growth are inevitable, they do not appear to be just around the corner.

2. AFS Background

The rationale, detailed design, and evolution of AFS have been well documented in previous papers [2, 5, 9, 10, 11, 15]. In this section, we only provide enough details of the current version of AFS (AFS-3) to make the rest of the paper understandable.

Using a set of trusted servers, AFS presents a location-transparent Unix file name space to clients. Files and directories are cached on the local disks of clients using a consistency mechanism based on *callbacks* [3]. Directories are cached in their entirety, while files are cached in 64 KB chunks. All updates to a file are propagated to its server upon `close`. Directory modifications are propagated immediately.

Backup, disk quota enforcement, and most other administrative operations in AFS operate on *volumes* [13]. A volume is a set of files and directories located on one server and forming a partial subtree of the shared name space. A typical installation has one volume per user, one or more volumes per project, and a number of volumes containing system software. The distribution of these volumes across servers is an administrative decision. Volumes that are frequently read but rarely modified (such as system binaries) may have read-only replicas at multiple servers to enhance availability and to evenly distribute server load.

AFS uses an *access list* mechanism for protection. The granularity of protection is an entire directory rather than individual files. Users may be members of *groups,* and access lists may specify rights for users and groups. Authentication relies on *Kerberos* [16].

AFS supports multiple administrative *cells,* each with its own servers, clients, system administrators and users. Each cell is a completely autonomous environment. But a federation of cells can cooperate in presenting users with a uniform, seamless file name space. The ability to decompose a distributed system into cells simplifies delegation of administrative responsibility [15].

As originally designed, AFS was intended for a LAN. However, the RPC protocol currently used in AFS has been designed to perform well both on LANs as well as on wide-area networks. In conjunction with the cell mechanism, this has made possible shared access to a common, world-wide file system distributed over nodes in many countries.

In 1990 the Advanced Research Projects Agency (ARPA) awarded Transarc a contract to deploy and

evaluate a file system to be shared by 40 to 50 Internet sites in the US. By mid-1991 there were 14 organizations included in the study. At the time of writing this paper, more than 80 organizations were part of this wide-area distributed file system (*wadfs*).

The wide-area nature of AFS is clearly visible from Figure 1, which shows the cells visible at the topmost level of AFS. All these directories, as well as the trees beneath them, are accessible via normal Unix file operations to any workstation anywhere in the system.

3. Evaluation Methodology

A comprehensive characterization of this system would include an assessment of basic architectural features, an analysis of quantitative data from the deployed system, and an examination of qualitative information reflecting on issues such as user perceptions of quality.

Since earlier papers have explored the architecture of AFS in detail, we omit it from this paper. Here we report on AFS from two angles: first, by instrumenting clients and servers and collecting data over a period of time; second, by circulating a questionnaire on various aspects of AFS to a sample of users and summarizing their responses. We believe that this combination of quantitative and qualitative information fairly characterizes the current state of the system.

One's confidence in the answers of an evaluation can be classified into four levels based on the origin of the information: *intrinsic* (direct examination of the system design), *empirical* (raw measurements), *evidentiary* (inferences based on raw data), and *anecdotal* (information requiring user judgment). In this taxonomy, our quantitative information is empirical and evidentiary while our qualitative information is anecdotal.

3.1. Quantitative Data

Empirical measurements of AFS were performed through the *xstat* data collection facility [17]. The AFS code was instrumented to allow collection of extended statistics concerning the operation of servers and clients. These statistics could be obtained remotely via an RPC call. A central data collection machine, located at Transarc, polled and obtained data from each participating machine four times a day. The collected data was formatted and inserted into a relational database for postprocessing. Figure 2 shows the structure of our data collection mechanism.

The scale of the system complicated the logistics of data collection considerably. It would have been practically infeasible to require the active cooperation of users or system administrators at many different cells to assist in the data collection. Hence our instrumentation required no regular administrative effort by the sites being monitored. However, the system administrator of a cell could turn off data gathering if that cell did not wish to participate in the study.

Not requiring the active cooperation of remote cells complicated the process of discovering which clients and servers should be contacted for data collection. Our solution to this problem was to run a discovery process once every few weeks. This process queried the Domain Name Service at each cell to obtain a list of registered IP addresses. This list was then probed to discover new AFS clients and servers in that cell.

The measurements were conducted during a 12-week data collection period from mid-May to mid-August 1993. Our data spans 50 file servers and 300 clients from 12 cells in 7 states. The only factors limiting broader coverage were the deadlines for this paper, and the need for participating sites to pick up the versions of AFS software incorporating our instrumentation.

cs.arizona.edu	theory.cornell.edu	soup.mit.edu	spc.uchicago.edu
cs.brown.edu	kiewit.dartmouth.edu	watch.mit.edu	ucop.edu
bu.edu	northstar.dartmouth.edu	ncat.edu	ni.umd.edu
cmu.edu	iastate.edu	eos.ncsu.edu	wam.umd.edu
andrew.cmu.edu	ucs.indiana.edu	nd.edu	umich.edu
club.cc.cmu.edu	isi.edu	nsf-centers.edu	citi.umich.edu
ce.cmu.edu	alefnull.mit.edu	pitt.edu	math.lsa.umich.edu
cs.cmu.edu	athena.mit.edu	psc.edu	lsa.umich.edu
ece.cmu.edu	rel-eng.athena.mit.edu	rose-hulman.edu	cs.unc.edu
sei.cmu.edu	media-lab.mit.edu	rpi.edu	css.cs.utah.edu
cs.cornell.edu	net.mit.edu	dsg.stanford.edu	cs.washington.edu
graphics.cornell.edu	sipb.mit.edu	ir.stanford.edu	

(a) educational cells

ads.com	ctp.se.ibm.com	prc.unisys.com	gr.osf.org
bstars.com	mtxinu.com	stars.reston.unisys.com	ri.osf.org
cards.com	locus.com	grand.central.org	syseng.osf.org
pub.nsa.hp.com	stars.com	ciesin.org	
palo_alto.hpl.hp.com	transarc.com	dce.osf.org	

(b) commercial cells

inel.gov	alw.nih.gov	ssc.gov	
nersc.gov	ctd.ornl.gov	cmf.nrl.navy.mil	

(c) goverment cells

jrc.flinders.oz.au	uni-freiburg.de	etl.go.jp	pegasus.cranfield.ac.uk
glade.yorku.ca	rus.uni-stuttgart.de	others.chalmers.se	athena.ox.ac.uk
writer.yorku.ca	sfc.keio.ac.jp	nada.kth.se	
lrz-muenchen.de	titech.ac.jp	bcc.ac.uk	

(d) cells outside US

This figure shows the cells visible from a typical client in the system. The listing above was obtained by doing an "`ls /afs`" and then sorting the output according to the domain. As the figure shows, there are 47 educational cells, 18 commercial, 6 governmental, and 14 cells outside the United States.

Figure 1: Cells visible from a typical AFS client.

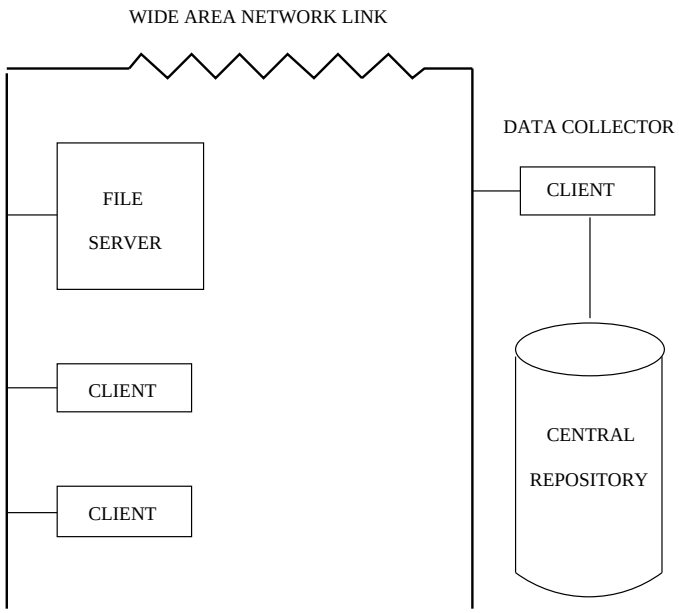


Figure 2: Instrumentation for Data Collection

3.2. Qualitative Data

To complement the quantitative data obtained by instrumentation, we constructed a questionnaire that touched upon a diverse set of issues. The purpose of the questionnaire was to elicit user perceptions as well as to obtain a profile of AFS usage. The topics of interest to us included characterization of the user community, extent of usage of native and foreign cells, and degree of collaboration within and across cells. We were also interested in obtaining user perceptions of performance and reliability of AFS for native and foreign cell access. Finally, we were interested in the value and adequacy of various AFS mechanisms such as access control lists, read-only replication, and data mobility.

The questionnaire was distributed in two ways: first, by posting on several Netnews bboards; second, by direct mailing to AFS contacts in different cells. We received about 100 responses from 50 cells. The data we present in this paper is averaged over all these responses.

4. Observations and Analysis

In this section we present both quantitative and qualitative data collected during our 12-week study. We begin by examining storage capacity and user profile. We then discuss the nature of client-server interaction, including RPC traffic and bulk data transfers. Next, we explore cache performance and availability, two key parameters of any distributed system. Finally, we examine the extent to which AFS is used for collaboration and information dissemination. In discussing these issues, we interleave the results of both empirical and anecdotal evidence, pointing out corroborations and contradictions wherever appropriate.

4.1. AFS Usage

4.1.1. Data Profile

Table 1 shows a recent snapshot of the data stored at 17 cells[2]. These cells comprised 95 file servers, housing almost 50,000 volumes and constituting over 300 GB of data. The data shows that although over

[2] These 17 cells were a superset of the 12 from which all other statistics in this paper are reported. We were able to obtain a larger sample in this case because the necessary instrumentation was present in an earlier release of AFS.

Volume type	Total	Size (GB)	Avg (MB/vol)
User	25,630	73	2.9
Backup	14,557	105	7.2
Readonly	2,121	24	11.4
Other	7,595	111	14.6
ALL	49,903	313	6.3

Table 1: Storage Capacities of 17 Cells

50% of the volumes belong to individual users, they contain only 23% (73 GB) of the data. A third of the data (over 100 GB) belongs to backup volumes. Only 4.2% of the volumes are readonly replicas, and they contain only 7.7% of the data. The remaining 15% of the volumes correspond to system binaries and data, bulletin boards, and other miscellaneous data. Together, these volumes contain one third of the total data.

Extrapolating from this evidence, and from additional information from the questionnaire, we estimate that the whole wadfs contains more than 200,000 volumes with 1.5-2 TB of data. It is interesting to note that although the average volume size is only 6.3MB, the raw data indicates that some volumes contain more than 1.5GB of data. In other words, volumes span a wide range of sizes but tend to be skewed toward the low end.

A related but distinct question pertains to how many of these volumes are in active use every day. To answer this question, we recorded the number of volumes whose activity level exceeded a specified threshold each day for the the duration of our data collection. The activity level was arbitrarily chosen to be 10 read references to a volume. Our data showed that, on average, a server has 65 active volumes, each containing about 16MB of data.

4.1.2. User Profile

The AFS user community consists of a number of academic, government and commercial sites and AFS users tend to have a very diverse background. However, responses to our questionnaire came mostly from AFS contacts, who are usually system administrators (Figure 3)[3]. The majority of respondents use AFS daily and for most of them the typical AFS session lasts a full working day. Most of them are serious programmers and two-thirds of them rate their knowledge of AFS to be at an advanced or expert level. Most of them had experience with other distributed file systems, usually NFS. Our sample thus represents a technically sophisticated group of respondents. This renders their assessments of AFS quality more credible, but also leaves unanswered the question of how naive users view AFS.

4.2. Client-Server Interaction Profile

How do AFS clients and servers interact? The answer to this question is important because knowledge of the relative distribution of file system RPC calls helps characterize a normal system and identifies the most common calls. This, in turn, allows performance tuning to be focused. Figure 4 lists the client-server RPC calls with short descriptions.

Both servers and clients have been instrumented to record the information regarding these calls. They keep statistics about the total number of calls, the number of successful calls and the average time of execution of successful calls (with the standard deviation). During our study, statistics were collected from 46 file servers and 264 clients on a typical day.

[3] The percentages for some questions do not add up to 100% because some respondents did not answer particular questions or they marked more than one choice.

1. What is your occupation?

9% Student
16% Researcher/Scientist
32% Software Developer
8% Manager
11% Support Staff
49% System Administrator
2% Other

2. How often do you typically use AFS?

0% Never
1% Rarely
11% Periodically
85% Daily

3. How long does your typical AFS session last?

6% Under 30 min
5% 30 min to 1 hr
12% 1 to 3 hr
74% Full working day

4. Which best describes the depth of your general computing experience?

1% Novice
15% Casual programmer
81% Serious programmer
2% Non-technical user

5. How would you rate your knowledge of AFS?

2% Novice
28% Intermediate
50% Advanced intermediate
17% Expert

6. What other distributed file systems have you worked with?

89% NFS
17% Apollo Domain
7% RFS
9% Other(s)

7. What's the best description of how you use AFS with the other file service resources at your site?

0% Don't use AFS at all
5% Use existing files in AFS, but none of my files are there
13% Store some of my files in AFS, most on other systems
14% Store many of my files in AFS
66% Most of my files are in AFS, including my home directory

Figure 3: A Profile of Survey Participants

`Fetch_Data`	Returns data of the specified file or directory and places a callback on it.
`Fetch_ACL`	Returns the content of the specified file's or directory's access control list.
`Fetch_Status`	Returns the status of the specified file or directory and places a callback on it.
`Store_Data`	Stores data of the specified file or directory and updates the callback.
`Store_ACL`	Stores the content of the specified file's or directory's access control list.
`Store_Status`	Stores the status of the specified file or directory and updates the callback.
`Remove_File`	Deletes the specified file.
`Create_File`	Creates a new file and places a callback on it.
`Rename`	Changes the name of a file or directory.
`Symlink`	Creates a symbolic link to a file or directory.
`Link`	Creates a hard link to a file.
`Make_Dir`	Creates a new directory.
`Remove_Dir`	Deletes the specified directory which must be empty.
`Set_Lock`	Locks the specified file or directory.
`Extend_Lock`	Extends a lock on the specified file or directory.
`Release_Lock`	Unlocks the specified file or directory.
`GiveUp_Call`	Specifies a file that a cache manages has flushed from its cache.
`Get_Vol_Info`	Returns the name(s) of servers that store the specified volume.
`Get_Vol_Status`	Returns the status information about the specified volume.
`Set_Vol_Status`	Modifies status information on the specified volume
`Get_Time`	Synchronizes the workstation clock and checks if servers are alive.
`Bulk_Status`	Same as `Fetch_Status` but for a list of files or directories.

Figure 4: Client-Server RPC Calls

Type of call	%	# of calls	(% err.)	Avg ms	(s.d.)
1. `Fetch_Data`	7.6	33,427,405	(0.2)	116	(486)
2. `Fetch_Status`	67.0	295,247,833	(18.0)	12	(378)
3. `Store_Data`	4.0	17,336,400	(1.0)	157	(744)
4. `Store_Status`	8.7	38,399,197	(0.3)	3	(119)
5. `Remove_File`	1.9	8,172,106	(0.0)	40	(335)
6. `Create_File`	2.0	8,945,032	(15.7)	22	(545)
7. `Extend_Lock`	1.8	7,815,294	(73.1)	9	(291)
8. `GiveUp_Call`	1.6	6,839,076	(0.0)	1	(39)
9. `Get_Time`	3.2	14,210,834	(0.0)	4	(800)
ALL	100.0	440,778,197	(13.8)	n/a	

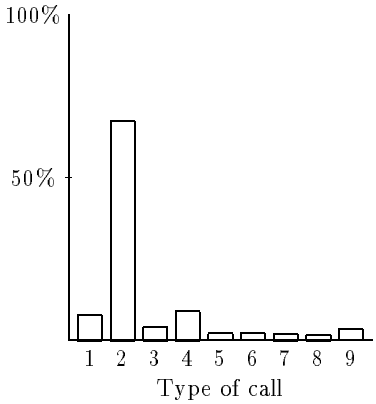


Table 2: Average Distribution of RPC Calls Observed by Servers

4.2.1. RPC Calls Observed by Servers

Over 440 million calls were observed during the data collection period. About 86% of these were successful. Table 2 summarizes the detailed statistics of calls accounting for at least 1% of the total.

The most frequent is `Fetch_Status` call. We conjecture that many of these calls are generated by users listing directories in parts of the file name space that they do not have cached. The relatively high number of unsuccessful calls (18%) suggests that these directories belong to some protected areas of the file name space. It is interesting to note that despite caching, the number of `Fetch_Data` calls is considerably higher than the number of `Store_Data` calls. Both `Fetch_Data` and `Store_Data` calls take considerably longer than other operations. This is to be expected, since they involve disk I/O.

`GiveUP_Call` turned out to be the call that takes the least amount of time on average. It was even faster than the `Get_Time` call, which is the simplest call. Considering the very high standard deviation of the `Get_Time` call, this might be just an anomaly in the collected data, but it can also be the result of a slow system call to get the time.

Although `Fetch_ACL` is not shown in Table 2, our raw data showed that it takes considerably more time on average than `Fetch_Status.` This surprised us, since `Fetch_Status` returns access list information. This apparent anomaly was explained when inspection of the AFS code showed that the implementation of `Fetch_ACL` contains a call to a protection server, while the implementation of `Fetch_Status` does not.

Analysis of RPC calls on a weekly basis confirms that their distribution is stable over time. Table 3 presents this data. This data shows only two significant deviations from the general profile shown in Table 2. One anomaly is the very high number of `Store_Status` calls during weeks 10 and 11. We discovered that more than 90% of these calls were concentrated on three file servers at Transarc. Further investigation revealed that these servers are frequently used for testing new AFS releases, thus explaining the unusual distribution of calls.

The second anomaly is the unusually high number of `Extend_Lock` calls during week 4. This is usually a rarely-occurring call, typically accounting for less than 1% of the calls in other weeks. Detailed analysis of week 4's data showed that the majority of these `Extend_Lock` calls were concentrated on just one file server. Our hypothesis is that there was a orphaned process on one of the clients repeatedly trying to make an `Extend_Lock` call, but failing because of expired authentication tickets. This also explains the high percentage of failed `Extend_Lock` calls in Table 2.

Based on this data, one can loosely characterize a normally running system as one with a very high

week	Fetch_D	Fetch_S	Store_D	Store_S	Remove_F	Create_F	Extend_L	GiveUp_C	Get_T
1	8.4	73.2	3.2	2.0	1.2	1.4	3.1	1.8	4.0
2	8.1	71.5	3.4	4.9	1.5	1.6	1.0	1.8	3.9
3	8.2	71.6	3.6	3.7	1.3	1.7	2.2	1.5	4.4
4	7.5	62.3	3.5	4.7	1.4	1.7	12.0	1.3	3.5
5	7.1	76.9	3.3	2.4	1.2	1.6	0.5	1.4	3.7
6	7.3	70.9	4.0	6.3	2.2	2.4	0.3	1.4	2.6
7	7.4	71.0	4.0	5.8	1.7	2.2	0.5	1.8	3.6
8	8.7	66.7	4.2	7.3	2.0	2.5	0.4	1.6	2.8
9	7.1	72.9	3.3	6.4	1.5	1.6	0.4	1.6	3.1
10	7.3	53.6	4.8	21.1	2.8	2.4	0.3	1.3	3.2
11	7.2	52.9	5.4	22.1	2.8	2.6	0.4	1.3	2.5
12	7.0	74.5	3.2	6.1	1.2	1.6	0.6	1.9	2.5
all	7.6	67.0	4.0	8.7	1.8	2.0	1.8	1.5	3.2

This table is based on the same raw data as Table 2. It indicates weekly averages (in percentages), rather than averaging across all weeks.

Table 3: Weekly RPC Call Distributions Observed by Servers

Type of call	%	# of calls	(% err.)	Avg *ms*	(s.d.)
1. `Fetch_Data`	7.6	9,141,014	(0.5)	158	(614)
2. `Fetch_Status`	54.4	65,450,704	(14.5)	56	(540)
3. `Store_Data`	17.3	20,816,713	(0.0)	65	(332)
4. `Store_Status`	9.8	11,806,214	(0.3)	30	(209)
5. `Remove_File`	1.0	1,260,655	(0.2)	61	(342)
6. `Create_File`	1.5	1,866,759	(12.6)	55	(642)
7. `GiveUp_Call`	2.2	2,680,433	(0.0)	65	(434)
8. `Get_Time`	3.5	4,188,167	(8.0)	33	(661)
ALL	100.0	120,192,852	(8.5)	n/a	

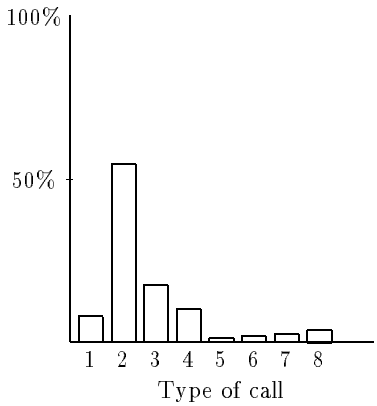


Table 4: Average Distribution of RPC Calls Generated by Clients

number (above 60%) of `Fetch_Status` calls, and smaller, but still significant, number of `Fetch_Data` and `Store_Status` calls (about 8%). Other frequent calls in such a system include `Store_Data` and `Get_Time`.

4.2.2. RPC Calls Generated by Clients

The set of machines from which we were collecting data did not represent a "closed system", i.e. there was no guarantee that participating servers and clients were contacting only each other. Thus, the number of calls observed by file servers does not match the number of calls generated by clients. Nevertheless, it is interesting to compare these two profiles. Table 4 summarizes the data collected from clients.

There were over 120 million calls, out of which 91.5% were successful. Again, `Fetch_Status` calls dominate. But the relative percentage of these calls was significantly lower than that reported in Table 2 for servers. At the same time, the relative percentage of `Store_Data` calls was significantly higher. Examination of the raw data showed that most of `Store_Data` calls came from a set of eight machines belonging to one cell. We conjecture that the applications on these machines differed substantially from the norm in their

	Servers		Clients	
	Fetched	Stored	Fetched	Stored
0 B - 128 B	32 %	44 %	33 %	6 %
128 B - 1 KB	4 %	7 %	5 %	15 %
1 KB - 8 KB	43 %	14 %	37 %	26 %
8 KB - 16 KB	4 %	6 %	4 %	8 %
16 KB - 32 KB	2 %	4 %	3 %	7 %
32 KB - 128 KB	14 %	25 %	17 %	7 %
over 128 KB	1 %	0 %	0 %	0 %
Daily per machine	156 MB	116 MB	5.3 MB	4.7 MB

Table 5: File Transfer Size Distribution

file access patterns. When these machines are excluded from the data set, the frequency of `Fetch_Status` calls increases to 62% and the frequency of `Store_Data` calls drops to 5%. The frequencies of other calls are similar to those reported in Table 2.

Surprisingly, Table 4 shows the average `Store_Data` call to be much faster than the average `Fetch_Data` call. It is even faster than the average `Fetch_Data` call on servers (Table 2), indicating negative network delay! This anomaly is also caused by the above-mentioned group of eight clients. When they are excluded from the analysis, the average time of `Store_Data` calls increases to a more credible $149ms$.

4.2.3. Causes of RPC Failures

As noted in the previous section, nearly 8.5% of the calls generated by clients failed. We were curious about the nature of these failures since they may have been symptomatic of underlying performance or reliability problems. To study this, AFS clients were instrumented to keep track of failed RPC calls. Errors were divided into several categories: server problems, network problems, protection problems (insufficient authorization or expired authorization tickets), volume problems, occurrences of a busy volume (e.g. when a volume is moved to another server) and errors of unknown cause.

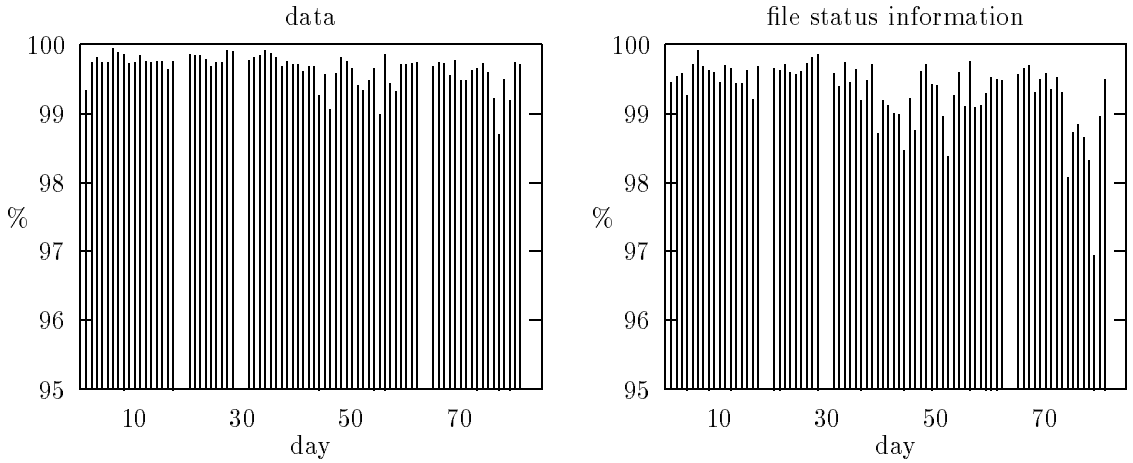
Our data showed that the majority of failed calls, 92%, were `Fetch_Status` calls. Most of them, 76%, failed because of protection errors. This is consistent with our earlier hypothesis of the existence of periodic jobs on some machines that attempt to traverse the AFS tree and fail when they encounter a protected part of the tree. Another plausible explanation is continuous execution of some background daemons (e.g. `xbiff`) which always produce a failed call after the authorization ticket's expiration. A significant number of unsuccessful calls, 22%, failed for unknown reasons.

4.2.4. Bulk Transfer Profile

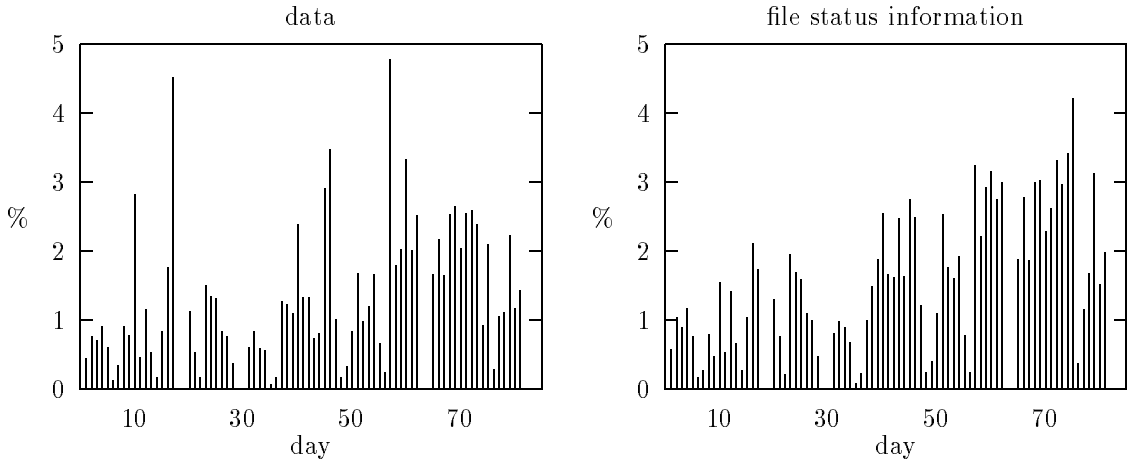
Statistics concerning file transfers were recorded by both file servers and clients. AFS performs partial file caching, so the numbers reported here show transfers on a per chunk basis, rather than on a per file basis. The exceptions are directories which are cached in their entirety. Chunk size is 64KB by default, but may be changed on a per-client basis.

The collected statistics are summarized in Table 5. Our data indicates that the most frequently fetched chunks are in the range 1-8KB. These correspond to entire files or directories. This result is consistent with many earlier studies of file size distributions which have reported small average file size [6, 8]. The second most frequently fetched chunk size is even smaller, in the range 0-128B.

The distribution of fetched data on file servers and clients is very similar. However, the distribution of stored data differs considerably. We can conclude that even when mixes of RPC calls and fetched data



(a) Combined cache hit ratio for native and foreign file references



(b) Fraction of references to files in foreign cells

This figure shows the observed cache hit ratios and relative proportion of native and foreign cell references over the data collection period. As explained in Section 4.3.1., data from six machines was excluded. Analysis of the raw data showed that the excluded machines exhibited comparable cache performance to the overall set of machines. The gaps in histograms on several days correspond to missing data due to problems with the data collection machine.

Figure 5: Cache Performance and Reference Mixes

8. How would you rate AFS performance when accessing files in your own cell (organization)?

22% Excellent
49% Good
20% Fair
4% Poor
3% Unsatisfactory

9. Compared to other distributed file systems you've used, is AFS in your own cell:

11% Much faster
30% Faster
32% Comparable
15% Slower
2% Much slower
5% Haven't used other distributed file systems

10. How would you rate AFS performance when accessing files in a remote cell?

8% Excellent
42% Good
38% Fair
6% Poor
2% Unsatisfactory
4% No experience

11. Have any of the following aspects of AFS ever seriously impeded your work?

65% Performance/reliability
21% Authentication/ACLs
6% Replication
19% Backup/restore
25% Semantics (Unix emulation)
32% Availability for other hardware/OS bases
10% Deployment (i.e., it doesn't run at the places with which I interact)
5% Other(s)

Figure 6: Users' Perception of AFS Performance

distributions are similar, there might be a significant variation in stored data distribution on servers and clients. The results from Section 4.1.1 indicate that the amount of data housed by active volumes is about 1GB per file server. Table 5 shows that only about 15% of this data (156MB) is actually fetched by clients.

4.3. AFS Performance

4.3.1. Cache Performance

Cache hit ratio is a critical factor in determining the overall performance of a system like AFS. Caching is especially valuable in masking the long latencies typical of wide-area networks. To study this aspect of AFS, clients were instrumented to keep statistics on cache hit rates and on the percentages of references made to native and foreign cells. Since the AFS file cache is split into a cache for data and a cache for status information, our statistics were kept separately for these two categories.

The overall percentage of references to remote files was 4.5% for data and 2.3% for status information. However, these numbers showed high variation from day to day: between 0.5% and 26% for data, and 0.5 and 34% for status. Closer inspection of the raw data revealed a group of six machines contributing to the majority of these references. We conjecture that these machines run periodic jobs that attempt to traverse the entire AFS tree [4]. Since these constitute pathological cases, we excluded these machines from our data set, and obtained the substantially more uniform results shown in Figure 5.

Our data indicates that the average cache hit ratio is over 98% for data and over 96% for status information. Over 95% of data and status references are to native cells. We statistically analyzed the possibility of foreign cell references causing much lower cache hit ratios. Our analysis indicated that there was no such correlation.

The responses to our questionnaire on AFS performance are presented in Figure 6. Most of the respondents rate the performance of AFS when accessing local data as good or excellent. Only 7% of users are not satisfied. AFS performance when accessing files in a remote cell is somewhat worse - 50% of respondents rate it as good or excellent, while 38% feel it is fair. Compared to other distributed systems they have used, 32% of respondents feel that AFS provides comparable performance, while 41% say that it is faster or much faster. Overall, the majority of users seem to be satisfied with AFS performance. But nearly two-thirds of

[4] This hypothesis has been verified for at least some of the machines.

Type of outage	% of time	Time (min/week)
Servers in the same cell	0.08 - 0.59%	1.2-8.5
Servers in the foreign cell	0.04 - 0.54%	0.5-7.7

This table shows observed average inconvenience times for clients over 12-week data collection period. The lower side of the range represents the case when for each client all daily failures occur simultaneously. The higher side of the range represents the case when daily failures do not overlap.

Table 6: Average Inconvenience Time for Clients

Downtime durations	Servers in the same cell	Servers in foreign cells
0 min - 10 min	1584	861
10 min - 30 min	759	128
30 min - 1 hr	484	67
1 hr - 2 hr	275	48
2 hr - 4 hr	140	21
4 hr - 8 hr	63	6
$>$ 8 hr	44	28
TOTAL	3349	1159

Table 7: Distribution of File Servers Outage Durations

them also rate performance and reliability as aspects of AFS that have sometimes been unsatisfactory.

4.3.2. Frequency of File Server Failures

Interruption of file service in a wadfs is a potential obstacle to providing transparency. One way of measuring file server downtimes is to have file servers record downtimes themselves and report them to the data collection agents. However, in our view, a much more important picture is the one that client machines have about the file servers' availability. Thus, we instrumented clients to record outages. A particular file server's downtime was observed only by the clients that could not access particular data from that file server (because of the server's failure and/or network problems). Such an approach weights failures by clients' interest in the files affected; in other words, the inaccessibility of a heavily-used file contributes more to the metric than the inaccessibility of a lightly-used file. Table 6 reports average *inconvenience* time, which is the time during which a client cannot communicate with at least one file server that it needs to access.

Downtime incident statistics were collected from 235 clients on an average day. During the twelve week data collection period, the number of observed server downtime incidents was 3349 for servers in the same cell and 1159 for servers in foreign cells. (Table 7). It should be noted that a particular server's outage can be reported multiple times if observed by multiple clients. Also, on an average day only about 15% of the contacted clients accessed data in foreign cells and thus were able to observe server downtimes in foreign cells. According to the numbers collected, on average, a client observes a server outage every 5-6 days for the local cell and every 3-4 days for the foreign cell, under the assumption that all clients are equally observant (active). The duration of almost half the outages is less than 10min. Since this is shorter than the recovery time for a typical server, we conjecture that many of these short outages are really due to transient network failures.

Users tend to notice file server failures less frequently than what the empirical evidence indicates (Figure 7). Failures of servers in local cells are experienced at most once a month by 77% of respondents. Only 3%

12. In your experience, how often are the AFS File Servers in your own cell (organization) down or unavailable?

2% Never
36% Once every few months
39% Once a month
17% Once a week
3% Daily

13. In your experience, for how long are AFS File Servers typically down when they crash or in the presence of network problems?

13% Less than 10 minutes
36% 10 minutes to 30 minutes
35% 30 minutes to 1 hour
12% More than 1 hour
3% N/A

14. In your experience, how often are the AFS File Servers in other cells (organizations) you access down or unavailable?

6% Never
20% Once every few months
29% Once a month
17% Once a week
7% Daily
20% N/A

Figure 7: Users' Perception of File Server Failures

Cells contacted	% of clients
≥ 1	100
≥ 2	67
≥ 3	42
≥ 6	22
≥ 10	15
≥ 20	9
≥ 50	4
≥ 70	3

Table 8: Client Contacts with Cells

witness file server failures on a daily basis. However, users perceive failures as lasting longer than the empirical data indicates: less than 10min for 13%, 10 - 30 min for 36%, 30min - 1hr for 35%, and more than 1 hr for 12% of respondents. Failures of servers in foreign cells are experienced at most once a month by 54% of respondent. However, the actual percentage is higher, because this question did not apply to 20% of respondents (question 14).

4.4. Sharing in AFS

The existence of cross-cell file access in AFS is borne out by the data presented in Figure 5(b). That figure showed that the percentage of references to the files in foreign cells was up to 5% for data and up to 4.5% for status information during the 12-week data collection period. Although 5% may not seem like much, it is significant because cells represent organizational boundaries and most users tend to access data within their own organizations.

Table 8 represents a histogram of the number of different cells contacted by each client during the 12-week period. The table shows that two thirds of the clients referenced data in at least one foreign cell while 3% of the clients referenced data in all available cells. Further, examination of the raw data shows that, on average, 15% of the clients referenced foreign data each day.

We also repeated the study originally reported by Kistler and Satyanarayanan [4] on the extent of sequential write sharing on directories and files. Every time a user modified an AFS directory or file, the user's identity was compared to that of the user who made the previous modification. Our data, showing that 99.1% of

15. What is the nature of AFS interaction between yourself and others in the same AFS cell? (Check any that apply.)

3% No interaction
39% "Looking around" your cell's file space to see what's new
69% Reading files
15% Accessing AFS-based bboards
45% Copying interesting files into your own storage area(s)
33% Copying files in one direction (e.g., drop-offs)
39% Copying files back and forth, modifying them at each step
66% In-place use, modifying files without copying them
6% Other

16. Have you ever explored/used the resources available through the grand.central.org cell?

34% Yes, I've used the materials there
26% Yes, I've looked through it to see what's there
21% No, I haven't had the need/desire
19% No, the /afs/grand.central.org directory has not been set up at my site

17. Have you ever explored/used the resources available at other cells?

71% Yes, I've used the materials at other cells
20% Yes, I've looked through other cells to see what's there
8% No, I haven't had the need/desire
1% No, the /afs/<other cell> directories have not been set up at my site

18. How many accounts/authentication identities do you have in other cells (i.e., how many cells other than your home cell can you klog to)?

20% 0
36% 1
19% 2
18% 3
7% More than 3

19. Rate the importance of the following communication/collaboration media and methods in your organization, using the following scale: 5: Very important, 4: Important, 3: Somewhat important, 2: Not important, 1: Not used at all.

3.92 Direct phone calls
2.43 Conference calls
2.55 Internal (paper) memoranda
2.14 U.S. mail
2.69 Express/overnight delivery services
3.26 FAX
2.66 Physical media (floppies, hard disks, mag tapes, etc.)
4.56 Email
3.49 BBoards/Email lists
3.55 FTP
3.06 Local (non-distributed) file system
3.38 Non-AFS distributed file system (e.g., NFS)
4.05 AFS
0.17 Other(s)

20. Are you currently working with people in another AFS cell on joint projects of any kind?

7% Yes, frequently
8% Yes, moderately
23% Yes, but not very frequently
15% No, the people I collaborate with outside my own cell do not (all) run AFS
43% No (for any other reason)

Figure 8: Users' Perception of Sharing in AFS

all directory modifications were by the previous writer, is consistent with Kistler and Satyanarayanan's observations. Unfortunately, we are not able to report on write sharing on files due to a bug in the statistics collection tools.

These observations confirm that the wide-area aspects of AFS are indeed valuable. Our anecdotal data, presented in Figure 8, corroborates this conclusion. Most users rate AFS very highly as a communication and collaboration tool. In their local cell, over 60% of the users tend to read or modify files that do not belong to them (question 15). Most users have used or looked at materials that reside in other cells. About 80% possess accounts/authentication identities in foreign cells. About 38% of the users participate in joint projects with people from different cells, although 23% do not do so frequently.

Further anecdotal information of the value of wide-area file access is provided by highly visible instances of information dissemination and collaboration in AFS. For example, AFS has facilitated the development of OSF's DCE. It has also been used in the STARS project initiated in 1990 by ARPA, which established a nationwide government-commercial collaboration. In both these cases, wide-area file access has been used by participant organizations to support sharing and dissemination. Project software and documentation are located in AFS and collaboration via AFS has occurred on a regular basis. AFS has also been used as a tool for information dissemination. The release of MIT's X11R5 software is a good example. In September

1991, the X11R5 release was installed into the cell `grand.central.org` and all AFS sites were able to immediately browse and access the release without manual file transfers.

5. Conclusion

Our goals in conducting this study were to observe a wadfs in actual use and to characterize its usage profile. We were also interested in determining how well AFS worked at the current scale of the system, and to see if any imminent limits to its further growth were apparent.

The qualitative and quantitative data that we have presented confirms that AFS provides robust and efficient distributed file access in its present world-wide configuration. The caching mechanism is able to satisfy most of the file references from the clients' local cache. Even though file server and network outages can be disruptive for particular users, our observations show that prolonged server inaccessibility is rare. Our data shows no obvious bottlenecks that might preclude further growth of the system.

AFS's divide and conquer technique of using semi-autonomous cells for spanning widely disparate organizations has proven to be invaluable. By providing considerable flexibility in security and storage management policies, the cell mechanism reduces the psychological barrier to entry of new organizations. As a consequence, growth in AFS over time has not just been in the number of nodes in each cell, but also in the total number of cells.

In summary, this paper provides conclusive evidence that AFS is a viable design point in the space of wide-area distributed file system designs. We are convinced that any alternative design must preserve the aggressive caching policies and support for autonomous administration that are the hallmarks of AFS' approach. The absence of either of these features will be fatal in any attempt to build a file system that uses a wide-area network and spans many organizations.

Acknowledgments

The *xstat* data collection facility was designed and implemented by Ed Zayas. Contributions to the evaluation methodology for wide-area distributed file systems were made by Ed Zayas, Alfred Spector and Bob Sidebotham. Anne Jane Gray provided assistance in organizing this project. Comments by Mike Kazar, Maria Ebling, Qi Lu, and Jay Kistler were helpful in improving the presentation.

References

[1] Baker, M.G., Hartman, J.H., Kupfer, M.D., Shirriff, K.W., Ousterhout, J.K., *Measurements of a Distributed File System.* Proceedings of the Thirteenth ACM Symposium on Operating System Principles, Pacific Grove, CA, October 1991.

[2] 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 Trans. on Computer Systems, Vol. 6, No. 1, February 1988.

[3] Kazar, M.L., *Synchronization and Caching Issues in the Andrew File System.* Usenix Conference Proceedings, Winter 1988.

[4] Kistler, J., Satyanarayanan, M., *Disconnected Operation in the Coda File System.* ACM Trans. on Computer Systems, Vol. 10, No. 1, February 1992.

[5] 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, Vol. 29, No. 3, March 1986.

[6] Ousterhout, J., Da Costa, H., Harrison, D., Kunze, J., Kupfer, M., Thompson, J. *A Trace-Driven Analysis of the 4.2BSD File System.* Proceedings of the 10th ACM Symposium on Operating System Principles, December, 1985.

[7] Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B., *Design and Implementation of the Sun Network Filesystem.* Summer Usenix Conference Proceedings, 1985.

[8] Satyanarayanan, M., *A Study of File Sizes and Functional Lifetimes.* Proceedings of the 8th ACM Symposium on Operating System Principles, Asilomar, December 1981.

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

[10] Satyanarayanan, M., *Integrating Security in a Large Distributed System.* ACM Transactions on Computer Systems, Vol. 7, No. 3, August 1989.

[11] Satyanarayanan, M., *Scalable, Secure, and Highly Available Distributed File Access.* IEEE Computer, Vol. 23, N. 5, May 1990.

[12] Satyanarayanan, M., *The Influence of Scale on Distributed File System Design.* IEEE Transactions on Software Engineering, Vol. 18, No. 1, January 1992.

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

[14] Spector, A.Z., *Thoughts on Large Distributed File Systems.* Proc. of the German National Computer Conference, October 1986.

[15] Spector, A.Z., Kazar, M.L., *Wide Area File Service and the AFS Experimental System.* Unix Review, Vol. 7, No. 3, March 1989.

[16] Steiner, J.G., Neuman, C., Schiller, J.I., *Kerberos: An Authentication Service for Open Network Systems.* Usenix Conference Proceedings, Winter 1988.

[17] Transarc Corporation, *AFS 3.1 Programmer's Reference Manual.* FS-00-D180, Pittsburgh, PA, October 1991.

Mirjana Spasojevic received the B.S. degree in mathematics from the University of Belgrade in 1986, and the M.S. and Ph.D. degrees in computer science from The Pennsylvania State University, in 1989 and 1991, respectively. She is currently working as a System Designer at Transarc Corporation. Prior to joining Transarc, she was an Assistant Professor at the School of Electrical Engineering and Computer Science, Washington State University. Her research interests include distributed operating systems and data management.

Mahadev Satyanarayanan is an Associate Professor of Computer Science at Carnegie Mellon University. He is currently investigating the connectivity and resource constraints of mobile computing in the context of the Coda File System. Prior to his work on Coda, he was a principal architect and implementor of the Andrew File System. Satyanarayanan received the PhD in Computer Science from Carnegie Mellon University in 1983, after a Bachelor's degree in Electrical Engineering and a Master's degree in Computer Science from the Indian Institute of Technology, Madras. He is a member of the ACM, IEEE, Sigma Xi, and Usenix, and has been a consultant to industry and government.