Inter-Office Memorandum

To	IFS Project	Date	November 10, 1977
From	Ed Taft and David Boggs	Location	Palo Alto
Subject	IFS Status and Plans	Organization	PARC/CSL

XEROX

Filed on: <IFS>StatusAndPlans.bravo

IFS is now at a stage in which it is available as a limited service, with limited capabilities and reasonably good reliability. It is now time to pause and consider how much additional effort should be invested in the project, and in what areas.

We recently conducted a review of the project, and our tentative decisions are presented here also. Members of the review committee were Chuck Geschke, Butler Lampson, Severo Ornstein, and Mike Schroeder.

Present Status

As a file server, IFS now supports most of the facilities available on Maxc. Exceptions include file protections, direct shipment of files for printing, and various odds and ends. With completion of the Backup system and Scavenger, file storage is now quite reliable.

The most serious shortcoming of the present system is its performance. We presently permit only five simultaneous users, and even with that load the system's performance is often very poor. Until recently the system crashed occasionally due to deadlocks arising from running out of memory.

We have not yet made a detailed analysis of the performance bottlenecks, but we have a good idea of where the problems lie. Unfortunately, they are not amenable to straightforward solution.

The Alto's main memory is divided into three regions: resident code, resident storage, and paging buffers for the software virtual memory. At present there are 32 1024-word paging buffers, which are used for a variety of purposes: code overlays (of which there were 61 at last count), server stacks (one per FTP or Chat user), stream buffers (one per file actively being transferred), B-Tree pages, large blocks allocated temporarily for scratch use, and several others.

The manner in which the Bcpl stack and overlay mechanisms work exacerbates the main memory shortage. Server stacks are allocated contiguously and can grow quite large (nearly 1024 words each). Once created, they cannot be moved or swapped out until the server is destroyed. Similarly, a code overlay cannot be moved or swapped out if any of its procedures is pending on any process's stack. Obviously, if there are several processes doing different things, many overlays will be locked in memory and few paging buffers will be left for other purposes.

We have recently expended about two man-weeks of effort in an attempt to increase the number of paging buffers. The result has been an increase from 27 to 32, which we hope will reduce the likelihood of deadlocks but is otherwise unlikely to improve performance in a significant way. We have reached the point of diminishing returns in this effort. There is little prospect of gaining any more Alto memory for paging.

We have given some thought to the possibility of using an Alto-II with extended memory. (Indeed, it was this possibility that led to the redesign and generalization of the extended memory option.) There is no straightforward way of using the extended memory in a general manner from within the Bcpl environment. However, there are a few instances of objects (such as stream buffers) that are referenced in sufficiently limited and stylized ways that it may be possible to locate them in the alternate bank.

Another possible use of the extended memory is as a cache for the disk, particularly for code overlays. We believe that swapping of code overlays accounts for a substantial fraction (perhaps more than half) of the disk transfers during heavy load. While use of a cache will not directly alleviate the situations that lead to deadlocks, it may improve throughput sufficiently that the number of simultaneous users will become fewer. (On the other hand, this will encourage more usage, so it won't help in the long term.)

Severo Ornstein is arranging that CSL's one extended-memory Alto will go to the Juniper project (in exchange for one of theirs), and Howard Sturgis has agreed that we can share access to that machine to facilitate experiments. If the results are favorable, that machine might replace the Alto-I that now serves as the IFS machine.

In any event, the disappointing performance of IFS is the principal reason we have not yet taken the plunge to move common files (such as the Alto directory) from Maxc1 onto IFS. At this point, we believe it would be unwise to do so until performance has been improved significantly.

Our List of Things To Do

This section contains a fairly complete list of all the IFS-related work we have contemplated for the immediate future. As can be seen, it is rather formidable, so it is unlikely we will reach the end of it before either abandoning the project or turning it over to someone else. The ordering reflects our assessment of the relative importance of the various improvements.

1. A Low-Level File Access Protocol

The fundamental reason for the present performance problems is simply that we are trying to make the server machine do more than it has capacity for. Of course, we have felt all along that a lot of the present machinery (and in particular, everything having to do with the user interface) ought really to run in the user machine; our attitude has been that the present Chat interface to many of the file system functions is merely a stopgap measure while we design a better file access protocol. However, we failed to anticipate the performance problems that the interim arrangements have encountered.

For some time we have considered the design of a *Page-Level Access Protocol* that would provide remote access to files at a very low level. Implementation of this protocol by a server would be relatively inexpensive, so the server could support simultaneous access by large numbers of users. An existing example of such a protocol is the one used by the Woodstock File System.

A Page-Level Access Protocol called "Pine" has been developed as part of the Juniper project. An attempt has been made to design the protocol in such a way that it may be used to access conventional file systems, such as IFS, as well as Juniper.

Given the ability to access IFS using such a protocol, we would then be in a position to construct a subsystem (and a set of packages) that would run in the user's Alto in place of FTP and Chat. All of the user interface and a substantial portion of the remaining code that now runs in the IFS server would then be contained in the user machine, leaving the server with more capacity to perform its basic functions.

Besides reducing the amount of code contained in the server, we expect this style of access to relieve the server's burden in a second way. A Pine request may be thought of as a remote procedure call: the user machine issues a single request, the server processes the request and generates a response, and the interaction is finished. The amount of state information that the server must maintain between requests is relatively small. It consists of connection state (for reliable transmission and perhaps authentication) and open file information.

The important point is that given the remote procedure call organization, a single server process can service requests for all users, rather than requiring one server process per user as is the case for the present FTP and Chat servers. This will result in a significant savings in main memory. (Actually, it is likely that we will distribute the requests among several servers so that long-running operations, such as deleting files, will not needlessly delay service of simpler requests.)

We therefore consider the implementation of a Pine protocol in IFS to be our highest-priority task. Reinforcing this, we have already committed to providing a page-level interface for use by Officetalk; the deadline for this is sometime during the first quarter of next year.

This effort may be divided into several, relatively independent projects. First, we must complete the design of the protocol itself (or modify the Juniper Pine protocol to serve our needs). Second, we must implement the server process that maps incoming requests into calls on existing internal IFS primitives. This should be relatively straightforward because we have already organized the system primitives with precisely this application in mind. Finally, we must implement the program that runs in the user machine. This entails a considerable amount of work, depending on how ambitious we are to produce a pleasing user interface. (It doesn't really matter whether it is written in Bcpl or Mesa.)

Recommendation: We will proceed with this effort, in cooperation with the Juniper project. Will Crowther, who recently joined CSL, will participate in the protocol design and probably a Mesa implementation. This work will be done in such a way as to benefit both IFS and Juniper. Our hope is that the outcome will be a set of user programs compatible with IFS and Juniper, at least for the most common functions.

2. Inter-Machine Access

The existence of multiple IFSs operated by different organizations has led to the obvious problem of sharing files among users of different systems. Since this problem is among the ones considered by the Juniper project, it is inappropriate for us to expend too much effort on it in IFS. However, some workable short-term solution is required.

This problem appears to break down into two cases. First, there are a number of large collections of files that are so heavily used that they must be duplicated on each IFS. These include such things as Alto subsystems, the Mesa system and library, fonts, documents, and so forth. Some sort of automatic distribution and update mechanism is clearly called for. (This problem has already been considered at some length by people in SDD and XEOS, and an FTP-based update system is presently being constructed. It is somewhat cumbersome because the primitives provided through FTP are inadequate. However, we might be able to solicit help from these quarters.)

Second, there are files that individual users wish to share with others, some of whom might not be users of the same IFS. We believe that in the long term, the *attachment* capability of

the Distributed Mail System will serve this purpose. In the meantime, another mechanism is required.

At the moment, each IFS has a "guest" account whose password is fairly public. This has somewhat uncomfortable security implications, and it will break down as soon as file protections have been instituted.

Butler Lampson has suggested an alternative arrangement whereby a user of any IFS is authorized to use any other (though perhaps with reduced capabilities, e.g., no disk storage rights). The most straightforward way to implement this is probably to arrange that each user's account be distributed automatically from his "home" IFS to all others. In order for this to work, we must ensure that protection groups are assigned in a uniform way across all IFSs. The major drawbacks to this scheme are that it does not extend well and that it encroaches on the local control each organization exercises over its own IFS.

Recommendation: Continue to live with the present arrangements for distribution of common directories. One person at each IFS site will be responsible for maintenance of such directories.

We propose the following policy for individual access to foreign IFSs: any user who desires such access will be given an account (with zero page allocation) on the foreign IFS upon request to his local IFS administrator. The local IFS administrator will have the capability to create accounts on all IFSs. To prevent naming conflicts, accounts on foreign IFSs will have the form "*organization/user*", where *organization* is the name of the organization to which the user belongs (e.g., Parc, SDD) and *user* is the name by which the user is known on his home IFS.

We will reconsider these questions after the Pine effort has been completed.

3. File Protections

IFS already contains a file protection mechanism very similar to the one used in Tenex; however, we have not yet provided any external access to protections. We have roughly designed an external representation for file protections. Completing the implementation should not be difficult.

Recommendation: Defer until after Pine, at which point it will be possible to implement much of the protection mechanism in the user machine.

4. Mail Server

IFS will eventually have to support the Distributed Mail System. While the DMS project can continue for some time using only the Maxc mail server, it requires an IFS mail server before it can receive wide use.

Duplicating on IFS the portion of the mail server that is presently implemented on Maxc should not be much work (indeed, some of it has already been implemented). However, several planned extensions, including forwarding, expansion of full DMS names, and access to attachments, will require a great deal more time.

Recommendation: Defer implementation until after Pine. The DMS people have expressed a desire for a lower-level (perhaps Pine-based) Mail Transfer Protocol, because the current FTP-based one is too expensive to implement. We will continue to work closely with the DMS group in considering this question.

5. Printing

At present, when a document is stored on IFS, printing it requires that it first be retrieved to another machine (an Alto or Maxc) before being sent to a printing server. This is somewhat inconvenient.

It is relatively simple to arrange for IFS to ship pre-formatted Press files directly to a printer. Pre-formatted Ears files are somewhat more difficult to send; given the imminent demise of Ears, we should not expend any effort here.

Unformatted files (e.g., Bravo or plain text) require a great deal of processing to convert to Press format. Our current feeling is that providing any sort of formatting service on IFS would be a mistake. (On the other hand, it might make a great deal of sense to construct a dedicated service Alto that retrieved files from IFS, formatted them, and shipped them to a printer.)

Recommendation: Implement only the printing of pre-formatted Press files. This will be deferred until after Pine unless someone else volunteers to implement it sooner. (Bob Lansford and Rick Tiberi of XEOS have made such an offer.)

6. Archiving

Some arrangement for archiving files permanently will eventually be required, although we can postpone the need for this almost indefinitely by buying additional disk drives.

Our calculations show that archiving to T-300 disk packs rather than to tape is feasible (assuming archiving volumes no greater than presently occur on Maxc). The disk technology we are using in IFS is close to state-of-the-art whereas the tape technology is not. As a result, our cost per bit on disks is only about double that for tape, and the storage density and reliability are much greater.

The actual transfer of files between primary and archival storage requires only a trivial modification to the present IFS backup system. However, the illusion provided to the user will require a fair amount of careful design (we consider the Tenex archive system to be deficient in several ways).

Recommendation: Defer indefinitely (i.e., until we need it).

Aside: The Alto Magtape system has not been abandoned, just postponed until we (or somebody) can find time to complete it.

7. Miscellaneous Other Servers

Several services now provided by the Nova Gateways are candidates for implementation within IFS's significantly more friendly file system environment. All of these services require maintenance of large data bases, for which IFS is ideal. However, some care must be taken to ensure that providing such services does not significantly degrade IFS's overall performance.

DMS and other applications will soon require an improved name lookup facility (perhaps something along the lines of Crowther's Clearinghouse Protocol).

A CopyDisk server by means of which one could store and retrieve disk images would be a welcome addition. The most common use of this facility would be the maintenance of the "basic" Alto disks.

An IFS-based Alto boot server would be an improvement over the present Nova-based

arrangement. This would not only remove the limit on the number of different boot files available but would make the booting process a great deal faster (its present slowness is due entirely to deficiencies in the Nova file system).

Recommendation: Reconsider after the Pine effort is completed.

Conclusion

The consensus appears to be that we should undertake no further development efforts that apply only to IFS. However, efforts that also benefit Juniper or that advance CSL's communication research activities will continue.