-- --

The LITTLE WORK Project†

Peter Honeyman, Larry Huston, Jim Rees, and Dave Bachmann
First.Last@umich.edu

Center for Information Technology Integration
The University of Michigan
Ann Arbor

INTRODUCTION

The continuing micro-miniaturization of components is moving high-powered, microprocessor-based machines from the desktop, to the laptop, to notebook-sized, and now palmtop computers. While distinguished in their hardware technology, supporting software has not kept pace: the predominant operating system on such machines is MS-DOS, absent any support for distributed computing. With the LITTLE WORK project [1], we propose to close this gap in the information technology environment.

The goal of the LITTLE WORK project is to use off-the-shelf components to develop a mobile computing environment that is identical to the one we enjoy in our offices. As a start, we need to select the appropriate building blocks for our system. Some of the pieces we have identified as being critical are: the computer, the operating system, the file system, and communications.

The Computer

Even the ideal mobile computer necessarily makes certain compromises required by volume, mass, power, or user interface considerations. *E.g.*, while a 19 inch screen (or more) is *de rigueur* on the desktop, this is impractical for a mobile machine.

Typical off-the-shelf components in today's marketplace in fixed and portable computers are shown in the table below. The marketplace and other considerations dictate that we concentrate on Intel 80*x*86-based systems.

	Fixed	Portable
MIPS	20 – 50	2 – 10
Display	1000×1000	640×480
Disk	100 – 500 MB	20 – 100 MB
Memory	8 – 32 MB	1 – 16 MB
OS	UNIX	MS-DOS
Comms	10 Mbps	< 2 Mbps

Operating System

Since we run UNIX everywhere else, we figured to run UNIX on our mobile computers. The Mach operating system [2] offers a close match to Berkeley UNIX, which we run at the office, runs on the 80*x*86, and is available to us in source form.

As versions of BSD UNIX become more widely available on the 80*x*86, we may revisit this decision. But one of the things we really like about Mach is not present in other breeds of UNIX: the external paging file. On our BSD desktop machines, we typically dedicate 35M to swapping. But because Mach pages to a regular file, it's unnecessary to dedicate disk sectors to swapping. This greatly economizes on an important, scarce resource.

File System

A distributed file system (DFS) is essential for a machine with limited permanent storage. Furthermore, a DFS allows system administration to be centralized, so that DFS clients don't require disk backup. So our goal is to run primarily in a ‘‘dataless’’' mode, with only those files necessary

† To appear in *Proc. of the Third IEEE Workshop on Workstation Operating Systems,* Key Biscayne (April 1992).

to bootstrap the operating system residing on the local disk. The DFS is responsible for providing all other files. Although losing the bootstrap files would be inconvenient, dataless computing obviates any need for backing up the local disk. We check our portables through airport X-ray scanners without a second thought.

Another requirement of our file system is to support some form of disconnected operation, continuing to function in the absence of network connectivity. We want to rely on cached copies of files in the DFS, yet minimize negotiations with the file servers before we can access the files. A mobile computer is normally disconnected from the network, so we intend to limit our reliance on network access.

Coda [3, 4] would seem to be the obvious candidate, but we have a lot of experience with AFS [5], both as users and as developers, and all of our files currently live on AFS servers, so we decided to stick with AFS. However, the AFS cache consistency mechanisms rely heavily on communication between the client cache manager and the AFS servers. We are modifying the cache manager to ease these restrictions. This is discussed in more detail below.

Communications

If the mobile computer is to be a replacement for our desktop environment, we need some sort of network connectivity. This is necessary to support a DFS, to read and send mail, and use other other distributed services.

While great strides have been made in wireless technologies, they are still limited, either in range (as with spread-spectrum) or bandwidth (as with satellite links). Instead, we chose to rely on dialup services, land-based and cellular, for our network. While bandwidth is limited, we are finding ways to make a modem-based system fully functional.

WHERE WE ARE

We began developing our prototype machine in the Fall of 1991. We are currently using an IBM PS/2 model L40, which is based on a 20 Mhz 80386SX. It has a 640×480 VGA screen, which is just (barely) enough resolution to run the X window system, and holds 10M of RAM (expandable, at a cost of battery life). The 60M disk is big enough to accommodate a healthy AFS cache with plenty of room left over for bootstrap files, paging space, and local applications.

We run Mach 2.5 from CMU. This provides us with a UNIX environment that supports all the applications that we use at the office, such as `xterm`, `mh`, `sam`, `troff`, *etc.*

For data communications, we are using two types of modems. The first is a standard V.32*bis* modem, with serial interface rates up to 57.6 Kbps. But to free us from the tyranny of wall jacks, we also obtained an Outback cellular modem from Eagle Cross, Inc.

The Outback contains a 3W Motorola cellular phone, a Telebit CellBlazer modem, and a battery pack. Although half-duplex, the CellBlazer offers higher data rates than normal modulation on a cellular link — we have been totally unsuccessful in our attempts to establish high-speed, full-duplex connections over cellular phone.

Unfortunately the cellular modem is significantly slower, heavier, and more expensive to use than the land based modems. A comparison of cold-cache AFS fetch rates through modems is shown below. Differences arise according to file type because of V.42*bis* or MNP5 compression in the modems.

Type	Land-line	Cellular
compressed	1630	330
i386 a.out	2475	500
source code	3340	650
email	2935	570

AFS cold-cache fetch
in characters per second

We have worked hard to enable AFS (and its underlying RPC, called Rx [6]) to operate well in a modem-based network [7]. To economize on limited network bandwidth and substantial cellular phone charges, we compress Rx headers, inspired by Van Jacobsons's work with TCP/IP headers [8]. This generally gets us an additional 10% from the network.

One serious problem with AFS is the inability of Rx to handle congested networks — and placing a relatively slow SLIP link into an otherwise high-speed network is a sure prescription for congestion. To fix this problem, we carefully followed Van Jacobson's prescriptions for congestion avoidance and control [9]. This greatly increased our throughput by reducing unnecessary retransmissions. Further work needs to be done to make Rx perform better over a CellBlazer.

At this time, we are able to produce a fair approximation of the desktop work environment. We have run the prototype from hotel rooms and airport lounges across the country, and one of us even took AMTRAK across the country with a LITTLE WORK prototype and the Outback portable cellular modem.

The trick is to wait for the train to approach a metropolitan area, where cellular service is usually available, establish a SLIP connection, pull mail down to the local disk and ship out any outbound messages queued up on the local disk. (We use a distributed mail service based on a post-office protocol.) When between cities, the new mail can be read at leisure and new messages composed. It's actually pretty easy to get work done this way.

Because AFS is fairly network intensive, especially when the cache manager is first started, we tend to mount the AFS service only when we plan to work for an appreciable amount of time, say an hour or more. And we go to some lengths to assure a hot cache before hitting the road with a LITTLE WORK along.

THE NEXT STEPS

The biggest difference between our mobile environment and the fixed environment is the absence of distributed file service when the modem is not connected. This requires us to violate our basic rule of dataless clients. Currently, when we want to use a set of files on the mobile computer, we have two options. The first is to establish a network connection and access those files using AFS. The second method is to explicitly copy these files to our local disk, and restore them to AFS land at a later time.

Neither of these methods is entirely satisfactory. The former requires that we maintain network connectivity, while the latter requires that we manage volatile, and presumably valuable, information on the local disk. Our vision of LITTLE WORK would allow us to establish a network connection and warm up our cache, then tear down the network and use the cache as though the network were up. Aside from the obvious benefit of centralized management of files, this would allow us to approximately double the size of the local cache, since we would no longer need private copies of the applications we commonly use.

We are currently developing disconnected operation for LITTLE WORK. Our strategy is to take some of the optimistic cache consistency principles used in Coda and to sprinkle them into AFS.

The basic idea behind optimistic replica control is for the cache manager to pretend that things are fine even though it can't really know. When it must perform an operation that might require server mediation, *e.g.*, creating or modifying a file, the cache manager logs the request and associated parameters (in a regular UNIX file), and performs the operation. Later, after establishing a network connection with the file server, a reconnect manager can determine whether the operation was in conflict with another writer while disconnected. If not, the reconnect manager presents the operation to the file servers for handling. If a conflict does ensue, and it is not the sort of directory conflict that can be sidestepped, the reconnect manager will notify the user and let her make a decision about the appropriate course of action.

A significant difference between LITTLE WORK and Coda is that we do not want to modify the file servers in any way. One reason for this requirement is to allow our mobile environment to talk to any of the ever growing number of AFS servers, including those outside our administrative control. Therefore, our attention is focused on the client.‡

Coda, on the other hand, has much broader goals. In particular, Coda supports replicated servers, and treats disconnected operation as a special case of network partition. The Coda reconnect discipline engages the support of Coda servers when the partition is healed, *i.e.*, when a network connection is re-established.

As of this writing we have implemented some of the basic disconnected operations. File reads and writes are working; we are still working on the directory operations. We are also working on cache reconciliation upon the reconnection of the network.

We would like to add some other functionality to the underlying operating system. One of our goals is to support ''hibernation.''

In traditional UNIX systems the computers are left on almost continuously, so there is little need to

‡ Our Rx improvements have been sent back to Transarc so that they may be included in the standard AFS server products.

ensure quick restarts. However, in a mobile environment this is no longer the case: uptime is limited by the battery life, typically a couple of hours. Currently, an appreciable fraction of that time is consumed by system startup procedures.

Instead, we plan to add a means of telling the system to swap out all processes to disk, and to push the kernel data space out to a file. When power is restored, the system will be able to come up in the same state it was in before the shutdown. We plan to work on hibernation this summer.

SUMMARY

The underlying thesis of the LITTLE WORK project is that laptop computers are now capable of supporting the kind of distributed computing environments common in academia and industry. The LITTLE WORK prototype makes a powerful statement in what is achievable today. Furthermore, it positions us to take advantage of further enhancements in computing technology: better screens, denser memory and disks, digital cellular, head-up displays, data gloves, *etc*.

ACKNOWLEDGEMENTS

The LITTLE WORK project is partially supported by Telebit and IBM. Thanks, Mike. David Rosenthal greatly stimulated our thoughts.

REFERENCES

1. P. Honeyman, ''Taking a LITTLE WORK Along,'' CITI Tech. Rep. 91-5 (August, 1991). `citi.umich.edu:/usr/techreports/citi-tr-91-5.ps.Z`
2. Mike Accetta, Robert Baron, William Bolosky, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young, ''Mach: A New Kernel Foundation for UNIX Development,'' pp. 93–112 in *Summer 1986 USENIX Conf. Proc.*, Atlanta (July, 1986).
3. Mahadev Satyanarayanan, James J. Kistler, Puneet Kumar, Maria E. Okasaki, Ellen H. Siegel, and David C. Steere, ''Coda: A Highly Available File System for a Distributed Workstation Environment,'' *IEEE Transactions on Computers* **39**(4), pp. 447–459 (April, 1990).
4. Mahadev Satyanarayanan and James J. Kistler, ''Disconnected Operation in the Coda File System,'' *Proc. ACM SIGOPS '91*, Pacific Grove, CA, pp. 213–225 (October, 1991).
5. J.H. Howard, ''An Overview of the Andrew File System,'' pp. 23–26 in *Winter 1988 USENIX Conf. Proc.*, Dallas (February, 1988).
6. ''Rx: Extended Remote Procedure Call,'' *Proc. of the Nationwide File System Workshop*, Pittsburgh, Information Technology Center, Carnegie-Mellon University (August, 1988).
7. P. Honeyman, L.B. Huston, and D. Bachmann, ''The RX Hex,'' Manuscript in progress, contact the authors.
8. V. Jacobson, ''Compressing TCP/IP Headers for Low-Speed Serial Links,'' *Internet Request for Comments*, Menlo Park, CA(1145), Network Information Center, SRI International (February 1990).
9. V. Jacobson, ''Congestion Avoidance and Control,'' *Proc. ACM SIGCOMM '88*, Stanford, CA, pp. 314–329 (August 1988).