Experience with Unicos on the Cray X-MP

Dennis M. Ritchie

Bell Laboratories
Murray Hill, New Jersey 07974

In November 1985, AT&T Bell Laboratories installed a Cray X-MP/24 system at its site in Murray Hill, New Jersey. This machine replaced a venerable Cray 1 (serial number 15). The Cray 1 was owned by the Murray Hill computer center, and operated as a company-wide facility. The X-MP, on the other hand, is owned jointly by the computer center and the Information Sciences division of the Bell Labs research area. For the computer center, the new machine is, so far, merely an upgrade of existing equipment; for the research area, it is an entirely new acquisition.

For those unacquainted with Cray's naming scheme, 'X-MP/24' means that the machine has two processors, and four megawords (32 megabytes) of main memory. There are eight DD-49 disks. The controlling operating system is COS. However, Unicos, Cray's version of Unix®, runs using the 'guest operating system' or GOS, facility. The machine is split precisely in half; it is configured so that, in effect, COS and Unicos each get one processor, half the memory, and half the disks. The COS part is used by the Computer Center, and the Unicos part by the research area. I will discuss our experiences almost entirely from Research's viewpoint, that is, our experiences with the Unicos half of the machine.

Our intent in obtaining the X-MP was to improve the research environment by making supercomputing facilities easy to use, and to encourage the undertaking of problems that couldn't be considered with our current hardware (a collection of VAX 11/750s plus a few more powerful machines). Over the last few months, though, our work has concentrated on adapting the X-MP to our environment, and learning to live with it. Specifically, we have been trying to

1. Live through the growing pains of the early releases of Unicos.
2. Move some of our own software to the Cray, in particular parts of the operating system.
3. Build and debug the hardware and software necessary to connect to the network that already links our other machines.

Dealing with Unicos

The Unicos operating system proper, and its ordinary command software, seem to be a well-done porting job of Unix System V, and we have had few problems with it. There have been some bugs in new code written specifically for Unicos, for example in the new *listio* system call, but the reliability of the system has been good. As mentioned above, we run Unicos as a guest operating system under COS; the Computer Center operations staff reports that on only a couple of occasions has COS been disturbed by Unicos. This is true even though Cray overstates the immunity of COS to guest Unicos malfunctions. In fact, if Unicos generates a bad I/O request to the IOS, the IOS will halt instantly and crash the entire machine. In practice, though, this does not to matter much; unless tampered with (and we have been tampering) Unicos does not generate such requests.

The chief difficulties we encountered are with the C compiler, other parts of the translation process, and the libraries. Cray's C compiler, which is an instance of Johnson's PCC [1], was originally prepared at Bell Labs, but has since been heavily modified by Cray. It follows the usual Unix scheme of compiling to assembly language. We have found a variety of bugs in the compiler, but many of them have already been repaired in more recent releases. It also generates inefficient code for some constructions; this too is being worked on.

The problem with the assembler and loader, imported programs from COS, is that they are dinosaurs from an earlier era. They are remarkably slow. The assembler, for example, is only about seven times faster per line of input than the Unix assembler on the VAX 11/750. That a typical C program seems to generate about three times as many assembler lines on the Cray as on the VAX makes the situation worse: when handling the output of the C compiler for a given program, the Cray assembler takes half the number of CPU seconds as does a 750, which makes the Cray spectacularly ineffective on a cost basis.

There is a similar story for the loader. More important, the assembler and loader do not currently support external names longer than eight characters, and produce only rudimentary information for debugging. We have modified our debugger to read the loader map ordinarily intended for printing, but it is insufficient. Currently, object files do not support the Unix 'bss' notion that allows suppression of 0-initialized data, so they tend to be much larger than necessary. The loader has a notion of a 'module,' which in combination with the conventions used by the compiler, means that the name of a C source file must be unique in its first eight characters among all related source programs, including those in libraries.

In early releases, the libraries, which are a melange of the transported Unix libraries and COS libraries, were buggy in construction, not well partitioned, and produced confusing results and unnecessarily large object programs. (At one point, compilation of a trivial C program produced a 150 kilobyte program text.)

This situation has improved considerably with each release, and it is clear that Cray Research agrees that there is still much work to do. Indeed, one of the major drains on our time is the assimilation of the new and reworked software that CRI delivers at frequent intervals.

Importing our software

Besides importing favored versions of various commands, we have made changes in the operating system to support the stream I/O mechanism of Eighth Edition Unix [2]. This scheme replaces the traditional character device and network interface with a more flexible one. The work is straightforward, because its connections with the rest of the operating system are few, but it has become evident that redoing the work each time a new release appears is an annoying task. So far, we have not moved other Eighth Edition features like the file system changes that allow remote file systems [3], the /proc file system for debugging [4] or our interprocess communication [5]. In part this owes to lack of time, in part to realization that further changes will be much harder to reconcile with future Cray updates, because they are more extensive. Thus we have to decide whether to press ahead and accept the consequences, or to wait until the system seems to have stabilized somewhat. We might also attempt to press Cray into accepting some of our changes.

Network connections

All our machines were already using a Datakit® VCS network [6], so it was important to attach the X-MP as soon as possible in order to be able to communicate with it conveniently. Datakit is a modular wide-area network that consists of switching nodes connected by various kinds of trunks. Each node may have several host and terminal interface boards that plug into a backplane along with the trunk interfaces. Within AT&T, there is an extensive, rapidly growing, network consisting of many hundred hosts and thousands of terminals. The switch backplane transfers packets at 7.5 megabit/second, but as with all networks, the true transfer rate is much lower. On our VAXes, we see about 500 kilobit/s between user processes on different machines under optimal conditions.

Because we wanted the Cray machine to be hooked up rapidly, we built as simple an interface as possible. It connects to a 'low-speed' (50 megabit/s) CPU channel. To run fast, an interface board should do at least some of the Datakit protocol, but we avoided that in favor of simplicity. Similarly, we connected our interface to a CPU channel rather than to the IO subsystem, so that the existing C code that conducts the protocol could be ported, instead of having to deal with IOS programming.

The result is much as we expected, though with a few surprises. The software was in full operation, and the X-MP in communication, within a few hours after the network hardware was debugged enough to work acceptably. (To be fair, I should admit that it took longer than expected to debug the hardware, and the software was thoroughly exercised while trying to run with the partially-working hardware). More recent work has uncovered more subtle hardware bugs, and several problems in the handling of the CPU channel by COS. As of this writing, the network is usable but not as fast as hoped: about 270 kilobit/s between the X-MP and a VAX. A combination of factors limits its speed. First, a persistent difficulty in the COS interrupt handling delays notification of Unicos when network completions occur; second, small hardware buffers in the VAX interfaces force the XMP to use small packets, which increases the number of interrupts and overhead. We believe that tuning should allow further improvement, but for proper communication between fast machines, new interfaces will be necessary: new VAX interfaces will permit much larger packet sizes, a new Cray interface will offload processing.

The network services available on the X-MP are with one exception the same as those supported between our existing machines, namely remote login (with transparency of ‘ioctl’ calls across machines); remote execution of commands; and file transfers of various kinds. A remote file server runs on the X-MP and allows other machines in our laboratory to view the X-MP file system as an extension of their own. The facility that is not yet available is for the X-MP to mount other systems’ files; the necessary operating system changes have not been installed. (The reverse direction is easier because the server is an ordinary user program.)

Performance and usability

We have been more interested in making the X-MP accessible and usable than in measuring or improving performance, but we have a few observations to offer.

Writing an ordinary file, using large (16KB) buffers, runs above 7 megabyte/s with our disks, which have a raw speed of about 10 megabyte/s. Reading back the same file similarly achieves 7 megabyte/s. We consider this good performance. However, with the system we are using now, free-list fragmentation soon reduces this rate for real file systems. On the other hand, a new version of the system we have just installed uses a bit-map allocation scheme that promises to reduce fragmentation.

CRI admits that the use of the guest operating system (Unicos running under COS) slows down I/O and system calls somewhat. We have not taken the opportunity to measure degradation for ordinary I/O operations, but it is certain that response to interrupts on the low-speed CPU channel under the guest facility is significantly slower. The inherent additional delay is uncertain, because we are not yet convinced that all the COS bugs in handling this device have been found.

CRI promises that CPU-bound programs run at the same speed under Unicos as they do under COS. We have not tested this but see no reason to doubt the claim.

We have observed that ordinary Unix utilities written in C, for example dc and troff, often run about 27 times as fast as they do on the VAX 11/750; programs that make heavy use of floating-point arithmetic look much better. This is somewhat disappointing when the ratio of the machine costs is considered, though there are mitigating factors. The C compiler would benefit considerably from some simple improvements and optimizations. For example, the calling sequence takes nearly twice as much time as it needs to, there is no instruction scheduling (let alone vectorization), and many code sequences are visibly non-optimal. On the other hand, some of the most beneficial optimizations have already been done, for example placing local scalars in B and T registers when possible.

The most important complaints of our users revolve around mixups and botches in the arrangement of the libraries, and the compiler programs that use them, the slowness of the network, and the still utterly rudimentary debugging facilities.

As system developers, our most important frustration is coping with things beyond our control, including bugs in COS, assimilation of rapidly-changing new distributions from CRI, and uncertainty about CRI's intentions.

Despite the problems, and especially given the observation that Cray's first official release of X-MP Unicos software is, as of this writing, only a few weeks old and has been delivered to only two sites, CRI is making a creditable effort and is producing a good product. It seems certain that an X-MP is not yet, and possibly never will be, a cost-effective general-purpose Unix machine. That is, it is a waste of money to buy an X-MP to replace a group of smaller machines doing the traditional mix of text processing, compilations, and netnews reading that Unix systems often do. Too much expensive floating-point and vector hardware will lie unused. On the other hand, for those with problems for which supercomputer power is needed, the availability of the Unix environment will make life easier and more productive.

Future plans

Use of the Guest Operating System facility of COS has been a valuable, even essential part of our operation of the X-MP. It allows nearly complete operational separation of the continuing responsibilities of the local Computer Center to provide stable service to a large user community, and the experimental environment we in the research area need. In particular, the ability to reboot guest Unicos remotely, while leaving COS operations undisturbed, is hard to imagine giving up.

Nevertheless, there are significant costs to the arrangement. It introduces an extra layer of software that must be understood, at least in part, in order to do Unicos development. The strict partitioning between the machine halves means that one CPU is left idle if one of the two coexisting systems is momentarily empty, even if the other is busy; there is an artificial limit on the memory available for programs in each system. Finally, of course, the Computer Center is forced to run COS; they wish to convert to Unicos, as soon as it becomes sufficiently stable. Therefore, around the end of the year, we plan to convert from COS and guest Unicos to a unified, standalone Unicos operation.

This change will require some accommodation on everyone's part, and considerable technical work as well. For example, we need to install a scheduler that will distinguish between Research and Computer Center users and give the two classes an equal share of the machine when it is busy, and to take care of a myriad of operational details. Perhaps most important, we need to find an equivalent of the Guest facility that will support testing of new system kernels while not disturbing production use.

References

1. Johnson, S. C. 'A Portable Compiler: Theory and Practice.' Proc. 5 Symposium on Principles of Programming Languages, January, 1978.

2. Ritchie, D. M. 'A Stream Input-Output System.' BLTJ 63 #8, October 1984.

3. Killian, T. 'Processes as files.' Proc. Usenix Conference, Portland, 1985.

4. Cargill, T. 'The feel of pi.'

5. 'Interprocess communication in the Eighth Edition Unix System.' Proc. Usenix Conference, Portland, 1985.

6. Fraser, A. G. 'Datakit−A modular network for synchronous and asynchronous traffic,' Proc. Int. Conf. on Communication, Boston, MA (June 1979)