M.I.T. Laboratory for Computer Science — September 20, 1977

Computer Systems Research Division — Request for Comments No. 150

Distributed Computing at PRIME

by Warren Montgomery

This report describes a distributed computing system being built by PRIME computer company. The system is known as Computer Cells, and was described by Dr. D. Nelson at the recent Brown workshop on distributed systems. Although this system does not have the same goals as the LCS distributed systems project, many of the ideas seem relevant to our work.

1) <u>Motivation</u>

The economic structure of a computer manufacturer dictates that it can produce and market most effectively machines of a particular range in size, weight, cost, and power consumption, independent of their computing power. Thus as technology improves, a company should try to produce more powerful machines with the same size and cost as their previous offerings, rather than producing smaller, cheaper machines with the same computational power. This can be done either by using faster circuit technologies, or by introducing more parallelism. The use of parallelism further divides into pipelining, array processing, conventional multiprocessing, and loosely-coupled multiprocessing.

Of these alternatives, loosely-coupled multiprocessing offers the possibility of increasing performance without major hardware or software development. In addition, loosely-coupled multiprocessing allows the company to offer products with a wide performance range using the same machines. This strategy can be used in combination with any of the others (faster circuits or more parallelism) to achieve additional improvements.

Loosely-coupled multiprocessing also offers all of the advantages of distributed systems seen by CSR, although these are of secondary importance to PRIME at this time.

2) Goals

With these ideas in mind, PRIME is designing a distributed system with the following goals. The system will have greater computing power than is currently available with PRIME equipment, yet should look to the user like a single large machine. The system should be able to grow incrementally in computing power, memory, file storage, or communications interfaces, without modifications to user-level software. The interface to the system seen by a user application should be independent of how many processors are available (even if there is only one) or the physical attachments of I/O devices and file storage to individual processors. The system should be fail-soft, in that the failure of a processor may cause tasks that were executing on that processor to be restarted, but any application that does not need resources available only through the failed processor should be executable after the failure, possible at reduced performance. Thus the failure of a processor in this system should be much like the failure of a memory board in a more conventional system. The system may crash because of the failure, but should be rapidly restartable with reduced processing power.

3) Approach

Given that the individual processors in such a system are no faster than the processors currently offered by PRIME, in order to improve performance, a means must be available for the user to describe the parallelism in his application. The means chosen was to allow applications to be described by a "high-level data flow description". Each application consists of a number of small modules, each performing some simple processing step. These modules are interconnected with buffered pipelines. pipelines. This seems a natural model for many business applications, and was used as the basis for the Software Tools software developent system [1]. It is interesting to note that when this idea was presented to the engineering department at PRIME, many people were skeptical about the amount of parallelism in most uses of computers. The marketing department believed, however, that many business customers described their applications as small processing steps interconnected by pipelines, or shared files. These applications fit the model very well.

The execution environment seen by the applications programmer is one in which there are a large number of small processes interconnected by UNIX style pipes. Each process executes a single software module which accesses files and pipes through the I/O statements provided by the high-level language in which the module is coded. For example, a FORTRAN program would perform its I/O through READ and WRITE statements specifying certain logical units, which had been initially connected to files or pipelines. The module is completely ignorant of whether the I/O is going to files or to pipes. A number of standard modules to perform useful functions would be provided by PRIME.

The file system seen by the modules is global to the entire system, and each module is completely ignorant of whether the files that it uses are local to the machine on which it executes or remote. The file system controls the concurrent use of files by processes, by allowing the creator of a file to specify whether the file can have only one user, one writer or n readers, one writer and n readers, or n1 writers and n2 readers. It is not intended that files will be intensely shared.

In addition to producing the modules needed for his application, the application designer produces a map that describes the interconnections between modules. This map shows how the logical units that are read or written by the modules are connected to pipes or files. In addition, it shows which pipes require buffering in order to avoid deadlock.

The modules and the map form an adequate logical description of the computation to be performed. In order to achieve maximum efficiency, the operating system requires some further guidance in the assignment of processes to processors, and in deciding how much buffering to provide for the pipes. This guidance is provided by a system manager, who may not be the same person as the application designer, and who has some global knowledge of the applications that run on a system. He supplies a logical to physical schema for each application that serves as a guide to the operating system in assigning process to processors. The schema can express notions such as the optimal amount of buffering to provide for a pipe, or that certain processes would run more efficiently if run in the same processor as certain files, or other processes. The schema can also express the computational load presented by each process as a guide to the initial assignment of processes to processors.

With this information, and information about the current configuration, the operating system assigns the modules of each application to available processors and establishes the pipe and file connections. Processes are never moved from one processor to another while they are executing, but if a processor fails, all applications which used that processor are restarted, and their processes are re-assigned to other processors.

4) Implementation

This distributed system is being built on top of the current operating system used by PRIME. This operating system provides a segmented virtual memory environment that is more restricted than that of Multics, in that the number of segments is smaller and there are restrictions on which segments may be shared among processes. A program does not in general reference a file by making it a machine addressable segment, as is done on Multics, but instead uses calls to the operating system to read and write files. Each process has a number of file units, that can be used to read and write files. Some of these units are used for standard input and output to system modules, such as source, listing, and binary for compilers, while others are used to perform I/O through FORTRAN, COBOL, or other language read and write statements. The implementation of computer cells will involve the addition of pipes to the operating system, and the implementation of a configuration manager, which assigns processes to processors and initializes pipe attachments.

Pipes will be implemented as saved files that can be connected to file units and read or written just like ordinary files. An interim implementation of pipes, which may also be provided to the user, forces applications to use a different interface to pipes than they use to saved files, but may be more efficient.

The network-wide file system mentioned above has already been implemented. PRIME's file system is organized into "logical disks", each of which contains a complete hierarchy and resides on a single physical disk. The logical disks can be mounted and dismounted, just like logical volumes on Multics. Processes running on a particular machine can be given access to files on logical disks that are mounted on drives connected to other machines.

When a process tries to access a remote file, it is suspended and a description of the operation to be performed is forwarded to a privileged process known as the file access manager (FAM). Each machine runs a FAM process which serves both as an agent for remote processes using files on that machine, and to request remote file operations from other FAMs for processes running on that machine. Thus when a remote reference is made, the local FAM negotiates with the remote FAM to perform the requested operation, and the local FAM uses privileged operations to copy data to or from the requesting process's address space.

Little work has been done on the configuration manager. It is expected that initially the logical to physical schema will contain a complete description of the assignments to be made, and that each application will have a number of schemas. The choice of which schema to use will be made based on which processors are currently functioning. A more elaborate configuration manager that makes process assignments more dynamically will be implemented eventually.

In order to allow processors to be added to such a system with minimal incremental cost, each processor must be capable of accessing all I/O devices, including its paging device or operator's terminal, through a communication network. Thus a new processor could share devices that were already present and attached to other processors. Preliminary study of the PRIME operating system suggests that adding this capability will be straightforward. Perliminary measurements on the load that this places on the network suggest that the network being used, which provides an 8 megabit effective data rate, is more than adequate.

5) Relevance to LCS

Although this system does not provide the kind of distributed computing environment proposed by LCS, the structure of this system forms an interesting semantic model for distributed computing. The notion of autonomous processes communicating through buffered pipelines appears to be a natural model for many applications of distributed computing.

I am currently working of a more precise description of this model and its relationship to others which have been proposed for distributed computing.

References

[1] Kernighan, B. W., and P. J. Plauger, Software Tools, Addison-Wesly, 1976.