



Fermi National Accelerator Laboratory

FERMILAB-Conf-92/260

Software for Parallel Processing Applications

Stephen Wolbers

*Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510*

October 1992

Presented at the *Computing in High Energy Physics Conference*,
Annecy, France, September 21-25, 1992

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Software for Parallel Processing Applications

Stephen Wolbers

Fermilab, Batavia IL 60510 USA

Parallel computing has been used to solve large computing problems in high-energy physics. Typical problems include offline event reconstruction, monte carlo event-generation and reconstruction, and lattice QCD calculations. Fermilab has extensive experience in parallel computing using CPS (cooperative processes software) and networked UNIX workstations for the loosely-coupled problems of event reconstruction and monte carlo generation and CANOPY and ACPMAPS for Lattice QCD. Both systems will be discussed. Parallel software has been developed by many other groups, both commercial and research-oriented. Examples include PVM, Express and network-Linda for workstation clusters and PCN and STRAND88 for more tightly-coupled machines.

1. Introduction

Computing problems in high-energy physics (HEP) and other scientific fields are often too large for standard conventional serial computing techniques. In addition, there are problems whose solution is not even attempted because of the large computing requirements involved. Parallel computing is a technique that can be used to provide cost-effective computing solutions to these large problems. This paper will motivate the need for parallel computing, describe the software and hardware used by Fermilab to provide such computing, review briefly other software packages available and finally discuss some possible future directions for this approach.

2. Motivation for Parallel Computing

Problems in HEP are quite large. Offline computing needs for a single run of an experiment typically range from 50 VUP-years (where VUP = one Vax 780 equivalent of performance) to 5,000 VUP-years or more. The typical experiment needs a few hundred VUP-years. These requirements are so large that standard mainframe techniques cannot be used in all cases to provide the computing solution within time and budget constraints. This problem has been recognized for many years as a problem that can be solved with parallel processing techniques [1]. The individual events are independent of each other and therefore can be passed out to individual processes, each of which runs a copy of the full reconstruction program. This is the standard farm approach to parallel computing. This technique has been pioneered at Fermilab with the ACP project and continues with CPS and UNIX farms.

A second HEP problem that requires large computing resources is Lattice QCD. A parallel computing solution is appropriate in this case because the problem requires the largest computing power available and is naturally parallel because of its grid-like nature. The software CANOPY [2] and the hardware ACPMAPS [3] at Fermilab have been used to provide parallel computing solutions to Lattice QCD calculations.

Other problems in HEP and science can certainly benefit from parallel computing for solutions. It is also the case that many problems previously thought impossible can be attempted once the computing is made available cheaply and conveniently. The increased computing has the effect of enabling new and better physics. For example experiments can benefit from larger monte carlo datasets (still a limitation in many physics results) as well as more detailed detector simulations. Having additional computing allows experiments to collect additional data and reconstruct it fully. Any other compute-intensive problems in HEP can benefit by having additional computing, including accelerator physics problems, theoretical calculations, data analysis of large

datasets, etc. It is clear that cost-effective parallel computing can have an impact on many areas of HEP.

3. CPS and CPS_BATCH

A software and hardware solution to the loosely-coupled computing problem of the offline reconstruction of HEP events are the packages CPS[4] and CPS_BATCH running on dedicated UNIX workstations (the Fermilab Farms). CPS is a package of software tools that allows a computational task to be distributed among many processes distributed on many processors. CPS itself does not in general limit the way a problem is split up to take advantage of parallel computing. CPS provides several sets of tools including remote subroutine calls, process synchronization and queuing, message passing and bulk data transfers. CPS is written in ANSI C and uses TCP/IP as its communication protocol, allowing it to run on many flavors of UNIX including IRIX, AIX, ULTRIX, etc.

A typical offline HEP code run on the Farms has the following characteristics. The events are read from serial media (8mm tape), the event is reconstructed using a large and complicated HEP code written in fortran and the results of the reconstruction are written out to serial media for further processing and analysis elsewhere. Mapping this to CPS is done by splitting the program into 3 pieces ('classes' in CPS language). The first class reads the input tape and sends them to the second class. The reconstruction is performed in one of many copies of this class of processes and then sends the reconstructed events to the third class of process. An average job will have one class 1 process, 8-30 class 2 processes and 1 class 3 process. This 3 class structure is natural for HEP offline reconstruction. It should be emphasized that the 3 class structure is not required by CPS, and the number of processes within each class is easily changed without code changes. Many other topologies are possible and CPS does not restrict the user from creative solutions to particular problems.

CPS_BATCH is a software toolkit to provide a batch system for multiple user and multiple processor systems. CPS itself does not restrict the topology of an individual job - it is quite easy to start up processes anywhere on the network where an account exists. The CPS_BATCH software provides allocation of resources (nodes, tapedrives, etc.), queuing of jobs, basic monitoring tools, tape-mounting software, user and manager control, and other features necessary for a large multi-user production environment. Though this aspect of parallel computing is often ignored, it is a very large part of the overall effort in providing a production environment. CPS_BATCH uses a resource manager called the Production Manager (PM).

4. CPS and CPS_BATCH on the Fermilab UNIX Farms

A. Hardware

CPS and CPS_BATCH have been in use for production (7 day/week, 24 hours per day) at Fermilab since early 1991. The UNIX Farms at Fermilab consist of approximately 100 Silicon Graphics (SGI) workstations (models 4D/25 and 4D/35) and approximately 100 IBM RS6000 workstations (models 320 and 320H) used as worker nodes. Each node contains a local disk for system and paging and swapping and is equipped with an adequate amount of memory (typically 16 MeG). The Farms will be augmented with 80 additional SGI workstations (IRIS Indigo) and 44 IBM workstations (model 220) within a few months. The total computing power available is approximately 5000 VUPS (soon to be 8500 VUPS). The CPU power of each machine is measured using a suite of HEP codes.

In addition to the worker nodes mentioned above a separate set of machines (I/O nodes) have been acquired to provide the connectivity to tape and disk. There are 6 IBM RS6000's (models 320 and 530) and 3 SGI (2-processor model 4D/420) for this purpose. Tape and disk are SCSI connected to these nodes. There are a total of 70 8mm tapedrives and 60 GBytes of disk storage spread across all the systems. Ethernet is used to connect the machines and the farms are subnetted with approximately 15 worker nodes and 1 I/O node per each subnet. This is a subfarm. Between 3 and 6

8mm tapedrives and 2-8 GBytes of disk are attached to each such subsystem. NFS is used to provide access to the disk across the entire subsystem.

B. Users and Production Systems

Many experiments at Fermilab either have run or are running jobs on the UNIX Farms. They are E665, E687, E706, E731, E760, E771, E789, E791, D0 and CDF. To allocate computing on the farms requires that the farm be divided into production systems. Each production system runs on a subfarm and has at least two processes on an IO node (for tape reading and writing) and one or two processes allowed per worker nodes up to the number of worker nodes on the subfarm. Production systems are used by CPS_BATCH when jobs are started to place processes on the proper hardware for the production system. This allows allocation of tapedrives, CPU, disk-space etc. to match the needs of the physics program and the needs of each user group. A group in full production runs on 3 or 4 production systems simultaneously.

The 10 groups that have converted their code to CPS have all been able to make the conversion easily. The modifications to a user code are fairly small and are usually accomplished in a matter of weeks (including testing and debugging). CPS itself has had certain features modified to satisfy the varying needs of performance for some of the users. CPS_BATCH has been the major weakness in the overall system performance for the users. A great deal of effort has gone into making the batch system more robust and making it easier to recover from the multiple failures inherent in this type of distributed hardware. As part of the overall performance users have been encouraged to checkpoint long jobs at intervals of approximately 1 hour so that failures only cause 1 hour of lost computation.

The ratio of CPU to I/O is a very important consideration in any parallel processing application. The more CPU intensive a job is (per I/O transfer) the easier it is to gain increased computing in a parallel system. An ideal situation is one in which each processor is busy 100% of the time doing computing (efficiency) and adding nodes adds exactly that node's computing power to the problem (speedup). HEP offline reconstruction has been measured to require between 500 and 2000 instructions/byte of data. The UNIX workstations used in the Farms each are rated at about 30 VUPS. The ethernet network can transfer data at up to 1 MByte/sec (theoretically). Each HEP event ranges in size from 2-3 KByte to 500 KByte. Additionally CPU is required to read and write and reformat data as well as to send it across the network.

In general it is not easy to predict exactly how well a CPS job will do in utilizing the resources of its production system. All of the above factors influence the total throughput. We have relied on empirical determinations to allocate production systems to users. In the best cases (Monte Carlo or offline codes transferring very large blocks of data) the speedup is linear up to > 15 nodes and the efficiency is above 90%. We have not made extensive measurements above 15 nodes to find out where the speedup would start falling below a linear increase with nodes. We have measured, using a special CPS test job, the throughput versus number of nodes as a function of the size of the typical data transfer. Two conclusions can be drawn from that study. First, CPS can saturate an ethernet segment and second, larger data transfers tend to provide better overall throughput.

The total amount of computing time used by experimental groups to do their event reconstruction has been steadily increasing as we add nodes to the farms and as we improve the CPS software and work on the overall robustness of the systems. Approximately 1000 VUP-months has been delivered to user applications in a month during the best months up to now. Much of the overall inefficiency has been driven by scheduling and physics reasons and not by CPS or CPS_BATCH failures. Three experiments have completed their reconstruction efforts on the farms and one of them (E760) reconstructed over 1 billion events using only a small fraction of the farms (very efficiently). CDF is able to keep up with the maximum data-taking rate in the 1992 data run on the Farms.

C. Management and Administration Tools

Many tools have been developed along with CPS and CPS_BATCH as well as standalone UNIX tools to help us understand how to run a large parallel computing system. An X-window based operating console has been developed to provide tape-mounting capability for CPS and CPS_BATCH applications. Other systems at Fermilab have found the interface to be so attractive that other non-Farm systems are also using or will be using the same system. Approximately 100 8mm tape mounts per day are handled by the operator console. CPS and CPS_BATCH have many utilities for monitoring and configure the systems. Most of these tools are being ported to X-windows to provide a uniform easy-to-use interface. Many UNIX system-management tools have been developed to make the system administration of 200 nodes easier to handle. Standalone programs to provide accounting of CPU-use, real-time monitoring of CPU use and other system attributes have also been developed. Many more tools are being developed to make the management and control of this large system possible. Standard UNIX tools are not sufficient for handling large clusters such as the Farms.

5. CANOPY and ACPMAPS at Fermilab

CANOPY and ACPMAPS are described in a separate contribution to this conference [5]. CANOPY and ACPMAPS were developed to solve a class of computation problems that include Lattice QCD (grid-oriented problems). CANOPY deals directly with grids, sites on the grids, field data associated with the sites, and tasks to be done for some set of sites on a grid. The software deals only with the abstractions listed above and not with details of how the data and work are distributed or how to get field data from remote nodes. The CANOPY program can be run on arbitrary numbers of processor nodes and can be moved to any system supporting the software.

ACPMAPS is a MIMD computer consisting of INTEL i860 processors (formerly Weitek 8032) connected via a backbone of crossbar switching crates. The design of the system optimizes communication for small data transfers to allow maximum use of the CPU available for the problem. The peak speed of the system is 50 GFLOPS. The system has been in production for many users (30-40) doing Lattice QCD calculations of heavy-light quark systems and charmonium states.

CANOPY and ACPMAPS also have associated software for scheduling, accounting and management. Though the nature of the user community and the problems to be solved are both quite different from the Farms it is still a challenge to provide the computing capability on a 24 hour/day 7 day/week basis so that the physics can be done.

6. Other parallel software packages

There are a large number of parallel programming techniques that try to take advantage of various hardware and software packages. Many of them involve more traditional approaches (such as vector computing) and will not be discussed here. The two types of techniques I will discuss are those concerned with the loosely-coupled workstation model and the grid-oriented tightly-coupled model. Many other approaches are covered in this conference and will not be repeated here.

A. Loosely-coupled computing

It has been mentioned that there are at least 100 software packages available for doing loosely-coupled parallel computing on clusters of UNIX workstations. Each package has involved substantial programming and system effort to produce and support. Most of these packages are documented poorly if at all, are supported to varying degrees and are to a large extent research efforts. The goal of many of these approaches is that of scavenging idle workstation cycles that would normally go unused. Due to the nature of the support and goals the packages are not always suitable for large production-oriented computing that require some amount of stable and available computing.

Three of the most commonly-mentioned packages are PVM [6] (and HeNCE), Express [7] and Linda [8]. PVM is a message-passing toolkit that allows users to run

code on a heterogenous collection of networked computers. It was developed jointly at Emory University, the University of Tennessee and Oak Ridge National Laboratory. The program is available free of charge over the network and is used by many users to run scientific code. The user is responsible for structuring the program in order to run it as a parallel program in PVM. HeNCE is an X-window based software environment built to assist users to port code to a parallel computing system using PVM. PVM does not contain a batch system as part of its tools. PVM contains graphical tools for examining performance and analyzing jobs. PVM is a research project with fairly large support at present though future support is not a given.

Express is another message-passing toolkit though the emphasis is slightly different. PVM and CPS both emphasize coarse-grain parallelism (subroutine level) while Express tends to emphasize more fine-grained (DO LOOP) parallelism. Many tools are available to utilize the features of Express. Debugging tools, graphical monitoring tools, and profiling tools all exist. An interesting tool within Express is ASPAR, an automatic parallelizer for FORTRAN. Express is a commercial package.

Linda (and network-Linda) uses a slightly different paradigm than the other packages. Linda allows message passing via a concept called tuples. Processes can read and write tuples in such a way that data is processed by whatever process matches the tuple template. This paradigm has advantages such as natural load-balancing since the tuples are processed by whatever processors are free and match the tuple. Linda also has debugging and graphical tools for analyzing jobs. Linda is a commercial package.

All of the packages could be used for HEP applications – there is nothing terribly special about HEP code except that it is very large and not terribly well structured or consistent. Express' automatic parallelizer would probably not do too well with HEP code. The major questions that a user should have before using these packages are what level of support (bug fixes, performance enhancements, debugging, etc.) can be expected, the amount of effort needed to port and maintain code in the new environment, and the ability of the package to support a full batch production system. None of the packages available is able to provide all the functionality needed for full production.

B. Tightly-coupled computing

Grid-oriented parallel computing is a very popular area of study and research in computer science. Massively-parallel computing is a very promising technique for producing the next generation of supercomputers (Teraflop machines). Taking advantage of all that CPU power requires new ideas and new software. Besides CANOPY other packages are PCN [9] and STRAND [10]. The idea in all of these packages is to structure the program in such a way that the parallelism can be implemented naturally. PCN is a package developed by Argonne National Labs and Caltech and supported by them. It is a research project to better understand parallel computing on a massively-parallel computing system (the INTEL DELTA) and to provide computing to scientific grid-oriented problems (such as global climate modeling). Many tools have been developed including a graphical profiling tool called UPSHOT. STRAND is a very similar package.

All of these packages are designed with the goal in mind of allowing the software to express the scientific computing problem in parallel terms independent of the hardware upon which the program will run. The hope is that by making such a separation code can be ported to many platforms so that the advances in computing due to faster machines can be exploited without extensive recoding.

7. Conclusions and Futures

Parallel programming has been a solution to many large computing problems. Software packages have been written to try to simplify the work needed to take advantage of parallel processing, both in the loosely-coupled case allowing exploitation of cheap UNIX workstations and in the high-performance tightly-coupled computing problem of exploiting massively-parallel computers. Fermilab has successfully provided

solutions to both sets of problems with CPS and CANOPY. Production parallel computing is a fact of life at Fermilab.

The situation is certainly not ideal. The plethora of packages and ideas indicates that the solutions invented and available are not adequate for everyone (maybe anyone). Ease of programming, debugging, monitoring, administration, allocation, resource management, load leveling, etc. are all in need of improvement. Porting code to parallel systems is never easy. Possible solutions may involve some synthesis of current ideas and packages into standardized tools or parallelizing compilers or some combination of these ideas. Tools such as HeNCE are promising in that they allow the casual user to compose parallel programs without learning how to insert parallel directives into the code by hand. Other approaches of a similar power would be very helpful and welcome.

More powerful tools for utilizing parallel processing can only improve the productivity of the people using computing to solve problems. The rapid advances in microprocessors promise further increases in computing power at falling prices. Hopefully communications will also see rapid gains. The future looks good for parallel computing in HEP.

Acknowledgments

I wish to acknowledge the work of the Fermilab Groups responsible for the success of the UNIX Farms: Matt Fausey, Bill Meyer, David Potter, Frank Rinaldo, Marilyn Schweitzer, Roberto Ullfig and Robert Yeager of the Farms Systems Group and Lisa Amedeo, David Fagan, Marc Mengel, David Oyler and Matt Wicks of the UNIX Systems Support Group. Many valuable contributions have come from Joel Butler, Chuck DeBaun, Brian Troemel, Paul Lebrun, Jim Meadows, Al Thomas, Mark Leininger, Frank Nagy, Eric Wicklund, Peter Cooper, Mike Diesburg and Tom Nash. I also wish to thank Al Geist of Oak Ridge National Labs and Ian Foster of Argonne National Labs for valuable discussions about PVM and PCN and parallel programming in general.

References

- [1] T. Nash, *Comp. Phys. Comm.* 57 (1989) 47.
- [2] Details of CANOPY can be found in the CANOPY 5.0 Manual, M. Fischler, G. Hockney, P. Mackenzie.
- [3] M. Fischler, "The ACPMAPS System - A Detailed Overview", FERMILAB-TM-1780, 1992.
- [4] M. Fausey, "CPS and the Fermilab Farms", FERMILAB-Conf-92/163, 1992.
- [5] M. Fischler, "Experiences with the ACPMAPS 50 GFLOP System", in these proceedings.
- [6] Al Geist, *et.al.*, "A User's Guide to PVM Parallel Virtual Machine", Oak Ridge National Laboratory/TM-11826, July, 1991.
- [7] Express, Parasoft Corp., 2500 East Foothill Blvd., Pasadena, CA 91107.
- [8] David Gelernter, *et.al.*, "Adventures with Network Linda", *Supercomputing Review*, October, 1990.
- [9] Ian Foster and Steven Tuecke, "Parallel Programming with PCN", Argonne National Laboratory ANL-91/32, December, 1991
- [10] Ian Foster and Stephen Taylor, *Strand - New Concepts in Parallel Programming*, Prentice-Hall, 1990.