Parallel Processing - Today and Tomorrow

E. McIntosh [a],

[a] *Physics Data Processing Group, Information Technology Divison, CERN, CH-1211 Geneva 23, Switzerland.*

A review of experience at CERN and elsewhere with Parallel Processing covering different Programming Models and current applications and platforms, is followed by some suggestions to facilitate the deployment and utilisation of large numbers of commodity processors.

Key words: application programming; commodity computing; parallelization; portability; system management; massive parallel systems

1 Introduction

Any HEP application is thought of as one task, and parallelism a complication. Yet there are many computer systems in the computer centre or on desk tops which could be exploited to economically provide the enormous capacity required for simulation or to provide a faster time to solution. Unfortunately the concept of parallelism has been hi-jacked and is associated with "Grand Challenge" problems, huge budgets, Teraflops, and with specialised and unbalanced systems. The natural parallelism, which arises from independent events, is scorned as being embarassingly or trivially parallel; yet a large number of HEP papers report on experience with workstation clusters and farms. Initially this may have just meant merging the outputs of a set of batch jobs, but is increasingly concerned with the use of explicitly parallel software like the Cooperative Processing System (CPS) at Fermilab or the Funnel system of DESY. The Centrally Operated Risc Environment (CORE) at CERN can be regarded as a metacomputer running many batch jobs in parallel with fast access to shared data.

The key issue is how to split up the work and combine the results. There is no compiler today capable of doing this effectively. Message passing is used but this is complicated and error-prone, and is not normally robust. There are applications where the organisational problems and the communications overheads render parallelisation ineffective over more than a few processors. Input/output over a network in loosely coupled systems is the main bottleneck in HEP and every effort must be made to improve it as with rfio in SHIFT at CERN and elsewhere.

The simultaneous availability of resources is a prerequisite for effective parallel processing; most scheduling and queuing systems are designed to share an expensive resource rather than guarantee the availability of dedicated resources or to use spare capacity.

2 The Applications

The principal parallel processing system at CERN is a Meiko (now Quadrics Supercomputers World Ltd) CS-2 installed as part of the GPMIMD2 Project (General Purpose Multiple Instruction Multiple Data 2) funded by the European Union. It is a 64 node system, with 128MB of memory, two processors and a local disk on every node [1]. A fat tree logarithmic Elan/Elite network at 50MB/second connects nodes and there are excellent system management facilities such as the partition manager with gang-scheduling, suspend/resume, and checkpoint/restart of parallel jobs. 800 GByte of external commodity SCSI disks are configured in Parallel File Systems (PFS). Single stream TCP/IP transfers through one FDDI interface operate at 6.5 MB/s (saturation at 12MB/sec) and a single job can read/write a scalable PFS on any other node at over 20MB/sec. The parallel run command prun runs n processes on N nodes of a given logical partition of the machine; multiple jobs may be active in a partition, using a subset of nodes, or time slicing, if adequate swap space is available.

The applications are in Fortran 77, C and C++. The model of full shared memory is little used; it is difficult to organise, to debug, and it is not portable to clusters. Using HPF or a multi-threading compiler slows down an application. The Meiko Atomic Transaction Library, which works well but is not portable has been used in PLATO (Perturbative Lattice Analysis and Tracking tOols) [2]. It provides an elegant solution to the problems of load balancing and task farming. All data is by default local but atomic transactions can be performed on programmer declared global data. It is easy to understand and debug. The most popular model is message passing, portable to clusters, PCs, and all parallel systems. Several applications use sockets over the internal network or just use multiple batch jobs accessing a shared data base.

The NA45 experiment performs event analysis using Objectivity databases stored in PFS. The NA48 experiment uses about half the CS-2 system for central data recording, parallel event simulation, and event analysis. [3]. A throughput of 20MB/sec from the experiment to tape is required and an attempt will be made to provide calibration data increasing the load on the disk buffer to over 60MB/sec. An event parallel version of GEANT 3 is being used by CMS, CPLEAR, DELPHI, NA48 and NOMAD. The NA48 simulations are special in that a shower library is first generated and then randomly accessed in parallel. The Energy Amplifier experiment PS211 uses up to 16 nodes for simulation studies with online monitoring of intermediate results. The Parallel Interactive Analysis Facility (PIAF) [4] has been moved from a dedicated workstation cluster to the CS-2. Two classic parallel applications are PLATO for accelerator design studies, and GRACE [5] a complete package for the automatic generation of Feynman diagrams. An investigation of the network characteristics and low level communications of the CS-2 and other systems was carried out by the RD11 project, concluding that they will be adequate in a few years time as second level triggers for LHC experiments [6].

The L3 experiment at CERN uses an adaptation of the ZEUS Funnel system FUL3, but has not ported it to the CS-2 or to any other Sparc architecture. Other experiments at other laboratories have developed solutions specific to their environment such as CPS or Condor. Condor and Funnel utilise otherwise idle resources for event simulation. CPS and SHIFT attempt to reduce I/O bottlenecks for event reconstruction and analysis. Finally, every DAQ system has an explicitly parallel architecture.

3 Trends and Conclusions

The performance improvement of RISC processors is confidently predicted to continue with Personal Computers providing yet another significant improvement in scalar price performance. The ATLAS experiment predicts a requirement of between 2500 and 5000 central processors, the larger number of slower processors being more cost effective. Cheap and effective solutions must be developed to the problems of parallel computing. The problems that need to be tackled are reliability, performance (I/O and networking), usability, portability and system management. Parallel systems like the CS-2, provide at least partial solutions to most of these problems, but are too expensive, and too late to market.

There must be no single point of failure in the system. All the task farming processes must be stateless and able to be restarted or replaced. Funnel attempts to do this but claims only that "loss of input events has no effect on the systematics of the simulation and only a negligible effect on its statistics". Even more important is data access. RAID Level 5 disk storage must be provided in hardware (if not too expensive) or in software. There must be a low latency and high bandwith network connection. Input and output can then be performed on remote nodes as with Condor, Funnel, or SHIFT. The system must operate across a heterogeneous mixture of shared memory systems, workstations and PCs running various UNIX or Windows flavours without operating system modification. Network wide tools for monitoring systems, job and task status are essential. Systems must be grouped in classes and characterised as being available for gang scheduling or cycle stealing.

The Network of Workstations (NOW) project at Berkely [7] is tackling these areas and is trying to improve remote I/O at reduced cost with Myricom, FDDI, ATM, and PCI interfaces. The NASA Beowulf system is trying parallel Ethernet channels. Hopefully from these research projects suitably priced networking capabilities will become available, thus solving the main performance and cost problem of today.

The HEP community needs to develop an architecture and programming model based on experience with CPS and Funnel, and to obtain global networked system management and scheduling tools. The NILE Fast-Track architecture [8] is a modern effort to do this with databases and object-oriented methods.

It is more important than ever to think parallel when designing new data handling systems, but to call it distributed computing or farming.

References

[1] E.McIntosh,B. Panzer-Steindel, Parallel Processing at CERN, *HEPiX96* (CASPUR, Rome, October, 1996).

[2] M. Giovannozzi, E. McIntosh, Development of Parallel Codes for the Study of Non-Linear Beam Dynamics (CERN/CN/96/17).

[3] F. Gagliardi, Remote Data-Recording and Processing for NA48, *Computing in High Energy Physics* (Rio de Janeiro,1995).

[4] T. Hakulinen, F. Rademakers, Experience of running PIAF on the CS-2 at CERN, in B. Herzberger and G. Serazzi,eds., *Lecture Notes in Computer Science no 919* (Springer, Berlin, 1995).

[5] F. Yuasa, D. Perret-Gallix, S. Kawabata and T. Ishikawa, PVM-GRACE, *Fifth International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics* (Lausanne, Switzerland, 2-5 September, 1996).

[6] R. Hauser, I. Legrand, A Real-Time Application for the CS-2, in B. Herzberger and G. Serazzi,eds., *Lecture Notes in Computer Science no 919* (Springer, Berlin, 1995).

[7] T.E. Anderson, D.E. Culler, and D.A. Patterson. A Case for NOW (Networks of Workstations), *IEEE Micro* (Feb.1994).

[8] D.G. Cassel and M. Ogg Nile: Distributed Computing and Databases for High Energy Physics, *NSF Grand and National Challenges Workshop* (March, 1996).