Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200017

Four Decades of Cluster Computing

Gerhard JOUBERT[a,1], Anthony MAEDER[b]
[a] *Clausthal University of Technology, Germany*
[b] *Flinders University, Adelaide, Australia*

Abstract.

During the latter half of the 1970s high performance computers (HPC) were constructed using specially designed and manufactured hardware. The preferred architectures were vector or array processors, as these allowed for high speed processing of a large class of scientific/engineering applications. Due to the high cost of the development and construction of such HPC systems, the number of available installations was limited. Researchers often had to apply for compute time on such systems and wait for weeks before being allowed access. Cheaper and more accessible HPC systems were thus in great need. The concept to construct high performance parallel computers with distributed Multiple Instruction Multiple Data (MIMD) architectures using standard off-the-shelf hardware promised the construction of affordable supercomputers. Considerable scepticism existed at the time about the feasibility that MIMD systems could offer significant increases in processing speeds. The reasons for this were due to Amdahl's Law, coupled with the overheads resulting from slow communication between nodes and the complex scheduling and synchronisation of parallel tasks. In order to investigate the potential of MIMD systems constructed with existing off-the-shelf hardware a first simple two processor system was constructed that finally became operational in 1979. In this paper aspects of this system and some of the results achieved are reviewed.

Keywords. MIMD parallel computer, cluster computer, parallel algorithms, speed-up, gain factor.

1. Introduction

During the 1960s and 1970s the solution of increasingly complex scientific problems resulted in a demand for more powerful computers. The available sequential processors proved unable to meet these demands. The attempts implemented in the late 1960s to optimise the execution of sequential program code by analysing program execution patterns resulted in optimised execution strategies [1, 2]. These attempts to increase the processing speeds of sequential SISD (Single Instruction Single Data) computers were limited and did not offer the compute power needed for the processing of compute intensive problems. A typical problem at the time was to be able to compute a 24 hour weather forecast in less than 24 hours.

A next step was to speed up the execution of compute intensive sections of a program through specially designed hardware. An often occurring operation in scientific computations is the processing of vectors and matrices. Such operations can be executed in parallel by SIMD (Single Instruction Multiple Data) processors. It was thus a natural approach in the 1970's to develop vector and array processors as the supercom-

[1] Lange-Feld-Str. 45, Hanover, Germany. E-mail: gerhard.joubert@tu-clausthal.de

puters of the day. Examples are the ICL DAP (Distributed Array Processor), ILLIAC, CRAY, etc.

The problem was that the development of such specially designed and built machines was expensive. The use of such supercomputers by researchers as well as software developers was limited due to the high cost of purchasing and running these systems. In addition the programming of applications software often had to resort to machine level instructions in order to utilise the particular hardware characteristics of the available machine.

The development of integrated circuits during the early 1970's, which enabled the large scale production of processors at ever lower cost, opened up the possibility to use such components to construct MIMD parallel computers at low cost. The concept proposed in a non-published talk in 1976 [3] was that the future of high performance computing at acceptable costs was possible by using standard COTS (Components Off The Shelf) to construct low-cost parallel computers. The architecture of such systems could be adapted by using standard as well as special compute nodes, different storage architectures and various interconnection networks.

The concept of developing such systems was, however, deemed unattractive during the late 1970's mainly due to two aspects. The first was Amdahl's Law [4] that only a relatively small percentage of programs could be parallelised, and the second was that the synchronisation and communication requirements would create an overhead, which made parallel systems highly inefficient. A further aspect that hampered the acceptance of MIMD systems, was Grosch's Law [5], which stated that computer performance increases as the square of the cost, i.e. if a computer costs twice as much one could expect it to be four times more powerful. This does not apply to MIMD systems as the addition of nodes results in a linear increase in compute power. Moore's Law [6] maintained in 1965 that the number of components per integrated circuit doubled every year, which was revised in 1975 to double every two years. This resulted in an estimated doubling of computer chip performance due to design improvements about every 18 months. It was an open question in how far these developments could offset the inherent disadvantages of MIMD systems.

In 1977 Prof. Tsutomu Hoshino and Prof. Kawai started a project in Japan to construct a parallel computer using standard components. Their aim was to develop a parallel system architecture that could be used to solve particular problems. The system was later called the PAX computer [7]. This approach was different from that described in the following sections, where the general applicability of MIMD systems to solve compute intensive problems was the main objective.

2. A Simple MIMD Parallel Computer

In 1976/77 a project was started at the University of Natal, South Africa to investigate the possibilities of achieving higher compute performances by connecting standard available mini-computers [8]. The final development stage was reached in 1979 when the system was upgraded to have both nodes with identical hardware. The parallel system was later named the CSUN (Computer System of the University of Natal) [8].

The project involved three aspects, viz. hardware and architecture, network and software.

2.1 Hardware and Architecture

The available hardware consisted of two standard HP1000 mini-computers. The processors were identical, but the memory sizes differed initially. The architecture decided on was a master-slave configuration with distributed memories. No commonly accessible memory was available. The HP1000 offered a micro programming capability, which allowed for special functions to be executed at high speed.

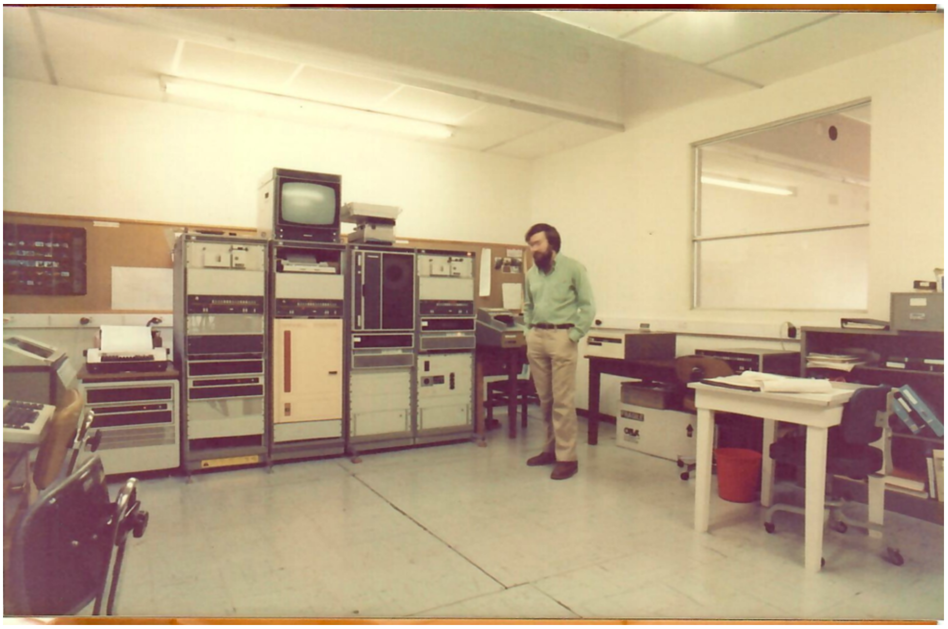

Fig. 1: The cluster system, admired by Chris Handley[2]

2.2 Network

The connection of the two nodes had to offer high communication speeds. This was realised by using a high-speed connection available for HP1000 mini computers for logging high volumes of data collected by scientific instruments. The cable was adapted by HP to supply a computer interface at both ends allowing the interconnection of the two nodes via interface cards installed in each machine. These interfaces were user configurable by means of adjustable switch settings for timing or logistic characteristics, allowing a computer-to-computer mode. The maximum transmission speed was one million 16 bit words per second.

2.3 Software

The Real Time Operating System (RTOS), HP-RTE, available for the HP1000 offered the basic platform for running and managing the nodes. The system had to be enhanced

[2] Later: University of Otago, New Zealand

by additional software modules to achieve the control of the overall parallel computer system. A monitor was developed to create an interface for users to input and run programs. Programs and data were provided on punched cards or tape.

A critical component was the communication between the two nodes. For this drivers were developed that also allowed for the synchronisation of tasks. With the master-slave organisation of the system the slave always had to be under control of the master. In an interrupt-driven environment this is easily accomplished. The communication available between the two nodes did not allow to transmit specific interrupt signals between the two machines. Thus data controlled transmission, i.e. sending all messages with header information, was used. Both sender and receiver had to wait for acknowledgement from the counterpart before message transmission could begin. This caused an additional overhead for the synchronisation of tasks.

The master node was responsible for all controlling activities. It prepared tasks for execution by the slave, downloaded these together with the data needed to the slave, which then started executing the tasks. The master in the meantime prepared its own tasks and executed these in parallel, exchanging intermediate results with the slave. The master also executed any serial tasks as required. The later upgrade of the system to have two equally equipped nodes simplified task scheduling.

Such a setup is of course very sensitive to the volume and frequency of data transmission. This must thus be considered by programmers when selecting an algorithm for solving a particular problem.

No programming tools for developing parallel software were available at the time. The standard programming language for scientific applications was FORTRAN. A precompiler was developed that processed instructions from programmers to automatically create parallel tasks that were inserted in the FORTRAN program code. The compiler subsequently created tasks that could be executed in parallel, which information was used to schedule the parallel execution of tasks.

3. Applications

The aim with the project was to show that at least some algorithms could be executed in less time by a cluster constructed with standard components. The two-node cluster was a starting point that could be easily expanded by adding more, not necessarily identical, nodes.

The physical limitations of the available nodes as well as the architecture of the cluster limited the classes of problems that could possibly be efficiently executed. Thus, a comparatively low volume of interprocessor data transfers as well as few synchronisation points relative to the amount of computational work, was an advantage.

Problems implemented on the cluster were, for example:

- Partial Differential Equations: One-dimensional heat equation solved by explicit and implicit difference methods [9]
- Solution of tridiagonal linear systems [10]
- Numerical integration [11].

4. Gain Factor

Several methods for assessing parallel computer performance are available, such as speedup, cost, etc. These metrics proved insufficient, especially in view of Amdahl's Law [4], for a comparison of the overall time used to solve a problem on a sequential processor and the MIMD system described above.

The measurement needed was a comparison of overall sequential compute time, T_s, and overall parallel compute time, T_p. A further aspect was that the optimal sequential and parallel algorithms may differ substantially. Thus, in the comparisons, the optimal algorithm for each processing mode—sequential or parallel—was used.

A large number of aspects influence the value of T_p, such as organisation and speed of processors (these need not be identical, thus potentially resulting in a heterogeneous system), interprocessor communication speed, communications software design, construction of algorithms, etc. In practice time measurements can be made to obtain values for T_s and T_p for particular algorithms. This gives a Gain Factor:

$$G = (T_s - T_p)/T_s$$

If $0 < G \leq 1$ parallel processing offers an advantage over sequential processing. The upper limit, $G = 1$, is obtained when T_p, the overall time used to solve a problem with the parallel machine, is zero. When $G \leq 0$ parallel computation offers no advantage. Note that G applies equally well to the performance measurement of heterogeneous systems, and includes communication and administration overheads and covers the limitations expressed in Amdahl's Law.

Results obtained for a number of test cases using the two node cluster, are [12]:

- Solution of tridiagonal linear systems, 120x120: $G = 0.42$
- One-dimensional diffusion equation, 30.000 time steps: $G = 0.481$
- Numerical integration, 30.000 steps: $G = 0.497$.

With a two node cluster the value of $G \leq 0.5$.

These results showed that, at least in some cases, parallel processing using an MIMD system with distributed memories may offer significant advantages.

5. Conclusions

The results obtained with the simple two-node MIMD parallel system showed that clusters constructed with standard components can be used to boost the execution of parallel algorithms for solving certain classes of problems.

The results obtained with the system prompted further research on the effects of more nodes, different connection networks and suitable algorithms.

This work resulted in the start of the international **Parallel Computing (ParCo) conference series** with the first conference held in 1983 in West-Berlin. The aims with these events was to stimulate research and development of all types of parallel systems, as it was clear from the outset that not one architecture is suitable for solving all problems.

It took more than a decade for the idea of using standard components to construct HPC systems to be adopted by industry on a comprehensive scale. It was also only gradually realised that the flexibility of cluster systems allowed for the processing of a wide range of compute intensive and/or large scale problems. The resulting advent of cheaper parallel systems built with commodity hardware lead to many specially designed HPC systems becoming less competitive due to their high price tags and limited application spectrum. The resulting major crisis in the supercomputing industry during the late 1980's and early 1990's lead to the demise of many companies supplying specially designed hardware aimed at particular problem classes..

Exascale computing is presently the next step in HPC and this will require extreme parallelism, employing many thousands or millions of nodes, to achieve its goals.

With the end of Moore's Law approaching, new technologies may emerge, to achieve the future development of HPC beyond exascale.

References

[1] Anderson, D. W., Sparacio, F. J., Tomasulo, R. M.: The IBM System/360 Model 91: Machine Philosophy and Instruction-Handling (1967), See: http://home.eng.iastate.edu/~zzhang/courses/cpre585-f04/reading/ibm67-anderson-360.pdf

[2] Schneck, Paul B.: The IBM 360-91, In: Supercomputer Architecture, The Kluwer International Series in Engineering and Computer Science (Parallel Processing and Fifth generation Computing), Springer, Bo ston, MA, Vol. 31, 53-98 (1987)

[3] Joubert, G.: Invited Talk, Helmut Schmidt University, Hamburg, January 1976

[4] Amdahl, Gene M.: Validity of the Single Processor Approach to Achieving Large-Scale Computing Cap abilities. AFIPS Conference Proceedings (30): 483–485. doi:10.1145/1465482.1465560, (1967)

[5] Grosch, H.R.J.: High Speed Arithmetic: The Digital Computer as a Research Tool, Journal of the Optical Society of America. **43** (4): 306–310 (1953). doi:10.1364/JOSA.43.000306

[6] Moore, Gordon: Cramming More Components onto Integrated Circuits, Electronics Magazine. **38** (8): 114–117 (1965)

[7] Hoshino, Tsutomu: PAX Computer, Reading, Massachusetts, etc.: Addison Wesley Publishing Company (1989)

[8] Proposed by U. Schendel, Free University of Berlin (1979)

[9] Joubert, G. R., Maeder, A. J.: An MIMD Parallel Computer System, Computer Physics Communications, Amsterdam: North Holland Publishing Company, Vol. 26, 253-257 (1982)

[10] Joubert, Gerhard, Maeder, Anthony: Solution of Differential Equations with a Simple Parallel Com puter, International Series on Numerical Mathematics (ISNM), Birkhäuser: Basel, Vol. 68,137-144 (1982)

[11] Joubert, G. R., Cloete, E.: The Solution of Tridiagonal Linear Systems with an MIMD Parallel Computer, ZAMM Zeitschrift für Angewandte Mathematik und mechanik, Vol. 65, 4, 383-385 (1985)

[12] Joubert, G. R., Maeder, A. J., Cloete, E.: Performance measurements of Parallel Numerical Algorithms on a Simple MIMD Computer, Proceedings of the the Seventh South African Symposium on Numerical Mathematics, Computer Science Department, University of Natal, Durban, ISBN 0 86980 264 X, 25-36 (1981)