GG

Preprint
DL-P-98-002

An Evaluation of Cost Effective Parallel Computers for CFD '98

D R Emerson K Maguire K Takeda and D Nicole

Submitted for Publication in Parallel CFD'98

5w9905

September 1998

The Central Laboratory of the Research Councils
Chadwick Library
Daresbury Laboratory
Daresbury
Warrington
Cheshire
WA4 4AD
Tel: 01925 603397 Fax: 01925 603195
E-mail library@dl.ac.uk

ISSN 1362-0193

Neither the Council nor The Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.

An Evaluation of Cost Effective Parallel Computers for CFD

D. R. Emerson^a, K. Maguire^a, K. Takeda^b and D. Nicole^b

^aCLRC Daresbury Laboratory, Daresbury, Warrington WA4 4AD, UK

^bSouthampton HPCI Centre, Department of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

An evaluation of the current status of two novel clusters, running Linux and NT, has been made to assess their performance for parallel CFD calculations. The core algorithm of the code is the conjugate gradient method used to solve the pressure equation. For good performance, a low latency and high bandwidth is required and the results indicate that, without substantial investment in this area, neither system is competitive on parallel performance when compared to more traditional distributed memory machines. At present, they are very suitable for development platforms and, for particular applications that are not too sensitive to latency and bandwidth, they offer a very cost effective computing resource.

1. INTRODUCTION

High performance computing using traditional vector computers, such as a Cray XMP or YMP, required a substantial financial investment from any organisation wishing to solve large scale computational problems. However, the move to building parallel computers that use commodity microprocessors changed this perspective and, whilst each processor was much less powerful than traditional vector processor performance, a machine could be built with a total performance that could rival and, in many instances, outperform traditional vector machines. This led to the notion of *cost-effective* computing. As off-the-shelf microprocessors were being employed, the cost of each system was much less than custom built systems. This trend continues today but it is now microprocessors from PC architectures, such as Intel's Pentium range, that are being seen as the new threat to conventional parallel computers such as Cray's T3E or IBM's SP2. The supercomputing market is relatively small (worth around \$3 billion) but the PC market is huge and is extremely competitive on price. This has led to continuous price decreases but with rapidly improving performance, as predicted by Gordon Moore many years ago.

A worrying concern, however, is that PC systems are primarily interested in integer performance whereas scientific calculations involve floating point operations. Whilst scalability is an important issue, the sustained performance will depend upon the single node performance. For applications such as CFD, which are very floating point intensive, this means that such systems may be a number of years away from being competitive on sustained performance. It should be remembered that processors used in current parallel machines, such as DEC's Alpha processor and IBM's RS6000, make use of superscalar technology, to overlap instructions and, combined with multiple floating point units (FPUs), produce more than one floating point operation per clock cycle. Many of these processors typically achieve about 10-15% of peak performance when not able to utilise scientific libraries (e.g. BLAS). Unfortunately, CFD is an application that is not generally able to make use of such libraries. This contrasts with traditional vector computers where a range of CFD applications have been shown to vectorise well and can often achieve 75% of the processor's peak performance. Whilst current generation PCs also make use of superscalar technology, they are, at present, limited to only one floating point operation per cycle and this will clearly have an impact upon their single node performance. This paper will therefore look at how effective current cluster technology is. The machines that will be considered include a DEC Alpha cluster running NT and a small Beowulf PC cluster running Linux. The two systems are typical of the many emerging offerings and their performance, both single node and parallel, will be compared to more traditional systems, such as the Cray T3E and IBM SP2. The code that will be used to evaluate these architectures is used in the direct numerical simulation of turbulent combustion. It uses a finite difference approach and most of the computational time is spent in solving a Poisson equation for the pressure field. A number of solvers are available for this, including multigrid, conjugate gradient and FFT.

2. DESCRIPTION OF ARCHITECTURES

The convergence of the high-end workstation and commodity personal computers now means that it is possible to build very cheap, powerful machines using commodity parts at a fraction of the cost of proprietary systems. The Beowulf initiative (<http://cesdis.gsfc.nasa.gov/beowulf/>) has concentrated on using Intel-based processors running Linux to provide very cost effective production machines for a number of applications. The system at Daresbury Laboratory is based on this approach where a number of Intel Pentium II processors have been purchased and the machine was assembled from scratch. The current system comprises of 2 dual-processors and 6 single processors operating at 266 MHz. Each processor has 128 MB memory giving a total aggregate memory of 1.28 GB and a peak performance of 2.6 Gflop/s. The operating system is Linux RedHat 5.0 and communication is via fast Ethernet with a peak transfer rate of 100 Mbit/s. The code is written in FORTRAN and the compiler used on the PC cluster was g77.

In terms of processors, the 64-bit Digital Alpha architecture currently offers very good price/performance in the low-end workstation and high-end PC markets. They are being priced to compete with Pentium-based systems and a cluster of 70 end-of-line Alpha workstations, Avalon [1], reached number 315 in the top 500 supercomputer list (www.top500.org) at a cost of \$150,000. However, increasing competition on Intel has forced them to release new chips ahead of their preferred schedule, making the Pentium II chip an attractive choice. This is based on the Pentium Pro, and therefore includes such advanced features as superscalar architecture and Dual Independent Bus architecture. Perhaps one of the most significant developments in this area is in the motherboard arena. PC bus clock speeds of 100 MHz are now common, increasing both Level 2 cache and main memory access speeds significantly. This highlights an important advantage of commodity supercomputers over proprietary systems - they can, in principle, be upgraded economically as new technology becomes available. The downside, however, is that the systems *date* very rapidly. The system at Southampton consists of eight 500 MHz DEC Alpha 21164 processors each having 256 MB memory giving a total aggregate memory of 2 GB and a peak performance of 8 Gflop/s. Two additional 5 GB drives allow the support of Windows NT 5.0, Linux Windows NT 4 (service pack three) and Linux RedHat 5.0 and communication is via a 100 Mbit/s Ethernet connection. The compilers available on the Alpha cluster were Egcs/g77, Digital Visual FORTRAN v5.0c and Visual C++ v5.0.

In this paper we discuss the code's performance only on commodity network hardware i.e. 100 Mbit/s switched fast Ethernet. Other, higher performance, proprietary systems are available, such as Myrinet (<http://www.myri.com/>) and Digital Memory Channel Interconnect [2], but their cost is high, of the order £1500 pounds per node at Q2 1998 prices. While these technologies do deliver good performance, they are not true commodity components and it may therefore be better to look towards technologies such as Gigabit Ethernet that will be coming available.

2.1. Operating Systems

An important issue when designing a commodity supercomputer system is which operating system to use. Linux provides a favourable environment, being powerful, POSIX-compatible, free, and with publicly available source code. NASA have been instrumental in developing so-called Beowolf-class systems that use mass market commodity off-the-shelf (M²COTS) PC components [3]. Recently, Red Hat has worked with NASA to produce Extreme Linux (www.readhat.com), a package containing the necessary drivers and additional software required to set up a Beowolf machine. While Linux provides a flexible, public-domain OS on which commodity supercomputing systems can be built, there is a growing move towards Microsoft Windows NT as an enterprise-wide strategy in many sectors. An advantage over commercial UNIX-based OS's is that it is practically free when purchased with new PC and workstation systems, due to Microsoft's

aggressive OEM licensing policies. Acquisition of Microsoft's FORTRAN Powerstation compiler and subsequent licensing of Microsoft Developer Studio by Digital means that Windows NT can offer a desirable, user-friendly development environment for scientific applications. Other vendors also offer FORTRAN compilers for NT with similar usability and integration with the MS Windows environment.

3. SINGLE PROCESSOR PERFORMANCE

The conjugate gradient solver used in the DNS combustion code makes use of level 1 BLAS calls (e.g. dot products and saxpy operations) if available. On the Cray T3D, for example, the level 1 BLAS routines sustain around 30 Mflop/s on large problem sizes and the equivalent FORTRAN routines deliver about 8 Mflop/s [4]. The performance of any application, however, depends upon the effectiveness of the compiler in optimising and exploiting architecture-specific enhancements. One advantage of using the Windows NT OS is that high performance compilers are available. In particular, the Digital Visual FORTRAN (DVF) compiler offers the same level of performance as the (more expensive) Digital UNIX version without the KAP optimising preprocessor. Recently, the KAP preprocessor has become available for DVF under NT and it is expected to bridge the slight (8%) floating-point performance gap in some benchmarks.

Figure1
Performance of Level 1 BLAS and FORTRAN routines on the DL PC processor

The results shown in figure 1 give the performance of two key sections of code in the conjugate gradient solver. This is for the dot product and saxpy operation. The results are all in 64-bit precision. The cache size of the Pentium processor is 16kB which is equivalent to 2048 elements (as indicated by the dotted line in figure 1). Some of the erratic behaviour is due to the granularity of the timing routine (updated every 0.01 seconds) but the figure clearly shows the expected drop in performance once the cache size has been exceeded. However, the saxpy (or daxpy in 64-bit precision) routine sustains about 19 Mflop/s and the FORTRAN about 12.5 Mflop/s, or between 5-7% of the peak performance. The BLAS libraries were supplied free (see www.cs.utk.edu/~ghenry/distrib/). The dot product, as expected, performs slightly better and the BLAS routine sustains around 30 Mflop/s (~11% of the peak performance).

Figure2
Performance of Level 1 BLAS and FORTRAN routines on the Southampton DEC Alpha processor

The results shown in figure 2 illustrate the single node performance of the DEC alpha processor used at Southampton. It has an 8kB direct mapped level 1 cache with a 96kB 3-way set associative level 2 cache and the effect of exceeding this limit is clearly seen. The tests were performed using Digital Visual FORTRAN v5.0c and the BLAS libraries were IMSL implementations. A range of compiler options were used to try to enhance the performance but from figure 2 it is evident that once the cache size has been exceeded these options have little or no effect. The BLAS routines show the better performance and, for the problem under consideration, it is not worth distinguishing between the various compiler options. It is the trend beyond the cache size that is important. Again, the granularity of the timing routine introduces some erratic behaviour for smaller problem sizes. The results in figure 2 clearly show that the floating point performance of this processor on these routines is very disappointing. The saxpy

operation sustains around 21 Mflop/s and the dot product about 23 Mflop/s. This represents a little over 2% of the peak performance of this processor.

4. PARALLEL PERFORMANCE

The performance of codes that use algorithms such as conjugate gradient methods and multigrid will critically depend upon the message passing bandwidth and, a very important factor, the latency [5]. Systems with a high latency do not, in general, perform well with multigrid algorithms because they have to communicate information about the residual on each mesh level. This involves short messages being sent very often. Another factor is concerned with the message size decreasing on each mesh level. For the conjugate gradient algorithm the frequent communication of global information, such as the dot product, also places demands on the latency of the system.

Table 1
MPI latency and bandwidth for cluster libraries and various machines

Machine	O/S	MPI Library	Latency (us)	Bandwidth (MB/s)
DEC	NT	WinMPICH	~2000	0.06
DL PC	Linux	LAM	150	8.5/(5.5)
DL PC	Linux	MPICH	200	8.5/(8.0)
T3D	UNICOS	Vendor	46	140/(30)
T3E	UNICOS/MR	Vendor	16	420/(170)
IBM	AIX	Vendor	80	38/(29)

At present, there are several message passing libraries available for Windows NT clusters. The three main implementations are: (i) High Performance Virtual Machine/Illinois Fast Messages (HPVM-MPI); (ii) WinMPICH 0.92 Beta [6] and MPI/Pro, produced by Mississippi State University and MPI Software Technology Inc.; and, (iii) WMPI32/PaTENT WMPI. Commercial variants of these libraries are becoming available and it is anticipated that the latency and bandwidth will improve dramatically as the market develops. At this point in time the only available MPI port for DEC Alpha machines running NT is WinMPICH, which was originally intended for SMP architectures and TCP support was only added at a later stage. The results for latency (in microseconds) and bandwidth are, therefore, not very good, as indicated in table 1. It should be stressed that these features are changing rapidly and the values presented represent what the current status is. Given the poor values the code predictably performs badly on the NT cluster and further results will be restricted to those on the PC cluster at Daresbury. This machine has two libraries available, WinMPICH and LAM, as indicated in table 1. The results are compared to a number of commercial systems. The bandwidth numbers were obtained for a message size of 10^5 64-bit elements. The number in brackets was obtained with standard MPI_SEND and

MPI_RECV calls and the other number represents the best obtained with asynchronous calls. At this point in time, the T3D and T3E clearly have superior performance but the gap between the cluster and the SP2 is not so large.

A comparison of the PC cluster with a Cray T3E/900 has been made to assess its relative performance. The timings shown in table 2 are for the conjugate gradient algorithm and have been broken down into the most time consuming parts of the routine. These include the time to perform all global summations, the preconditioning time, the message passing time, the sparse matrix operation, and the BLAS operations. These are shown in table 2 and are for a 96^3 problem running on 8 processors and all timings are in seconds. The first two operations shown involve message passing and the effect of latency (on the global sum) is very evident. However, the floating point calculations done in the sparse matrix and preconditioning operations both involve striding through the cache and the FORTRAN performance of the Pentium processor is very respectable. The BLAS routines all have a unit stride and the Pentium's performance is, again, very good and only a factor of 2-3 separates the results. Overall, there is only a factor of 4.5 in the performance and most of this is attributed to the latency and bandwidth. As the cost of the PC cluster was less than £15,000, this could be considered a very sound investment.

Table 2

Comparison of parallel performance on PC cluster and Cray T3E/900

Operation	T3E/900	PC cluster
global sum	1.19	52.75
message passing	11.2	172.65
preconditioning	12.75	35.94
sparse matrix ($p = Aq$)	27.46	64.82
BLAS	62.78	202
TOTAL	115.38	528.16

The scalability of the combustion code has also been tested on the PC cluster. These results, using LAM, are shown in table 3 for a range of problem sizes. The timings are in seconds and the speedup is shown in brackets. The 80^3 and 96^3 problems could not be run a single processor.

Table 3

Speedup of combustion code on Daresbury PC cluster using LAM

Number of Processors	$64^{**}3$	$80^{**}3$	$96^{**}3$
1	602 (1.00)	-	-
2	321 (1.88)	727 (1.00)	-
4	225 (2.68)	472 (1.54)	846 (1.00)
8	145 (4.15)	302 (2.41)	533 (1.59)

5. CONCLUDING REMARKS

Two novel computer clusters have been evaluated for a DNS combustion code which makes use of algorithms in common use in parallel CFD. The paper has investigated the true cost-effectiveness of commodity computing systems at this moment in time. The MPI message passing environment under Windows NT on DEC Alphas is, currently, quite immature. In reality, neither system is competitive on parallel performance when compared to more traditional parallel machines. However, it is recognised that more expensive offerings, such as that provided by Myrinet, will provide competitive latency and bandwidth and the code's performance would be improved substantially. It is also important to remember the very dynamic nature of this technology and the four principle areas of concern: hardware; O/S; compilers; and message passing libraries, have been discussed with an eye on future developments. For this type of cluster computing to become truly competitive, it requires all four areas to converge. At present, they are very suitable for development platforms and, for particular applications that are not too sensitive to latency and bandwidth, they offer a very cost effective computing resource.

REFERENCES

- [1] M. S. Warren, T. C. Germann, P. S. Lomdahl, D. M. Beazley, J. K. Salmon, "Avalon: An Alpha/Linux Cluster Achieves 10 Gflops for \$150k", Submission for 1998 Gordon Bell Price/Performance Price.
- [2] J V. Lawton, J. J. Brosnan, M. P. Doyle, S. D. O. Riordain, and T. G. Reddin, "Building a High-Performance Message-passing System for MEMORY CHANNEL Clusters", Digital Technical Journal, Volume 8, Number 2, 1996, pp. 96-116.
- [3] T. Sterling, T. Cwik, D. Becker, J. Salmon M. Warren and B. Nitzberg, "An Assessment of Beowolf-class Computing for NASA Requirements: Initial Findings from the first NASA Workshop on Beowolf-class Clustered Computing", In Proc. IEEE Aerospace Conference, March 21-28, Aspen, CO. 1998.
- [4] D. R. Emerson and R. S Cant, "Towards Direct Simulation of Turbulent Combustion on the Cray T3D - Initial Thoughts and Impressions", DL-TR-96-002, April 1996.
- [5] Y. F. Hu, D. R. Emerson and R. J. Blake, "The Communication Performance of the Cray T3D and its Effect on Iterative Solvers", Parallel Computing 22 (1996), pp 829-844.
- [6] S. Pakin, V. Karamcheti and A. A. Chien, "Fast Messages (FM): Efficient, Portable Communication for Workstation Clusters and Massively-Parallel Processors", IEEE Concurrency, vol. 5, no. 2, April-June 1997, pp. 60-73.