Monograph:
Cornish, I.J. and Zalzala, A.M.S. (1993) Transputer Benchmarks and Performance Comparisons with other Computing Machines. Research Report. ACSE Research Report 467 . Department of Automatic Control and Systems Engineering

---

TRANSPUTER BENCHMARKS AND PERFORMANCE COMPARISONS WITH OTHER COMPUTING MACHINES

Ian J. Cornish and Ali M. S. Zalzala

Research Report #467

February 1993

TRANSPUTER BENCHMARKS AND PERFORMANCE COMPARISONS WITH OTHER COMPUTING MACHINES

I. J. Cornish and A. M. S. Zalzala

Department of Automatic Control and Systems Engineering,
University of Sheffield,
P. O. BOX 600, Mappin Street, Sheffield S1 4DU, United Kingdom
Email: ali@uk.ac.sheffield

ABSTRACT

Since engaging in the actual implementation of a computer-based control application can be an expensive and time consuming procedure, engineering researchers need to assess their needs and choices before starting the job. This paper provides a performance index for a variety of hardware/software combinations to ease the decision making process. In addition, the transputer, as a single block in a multi-processor network, has been used successfully for a number of applications, where large computational power is needed. Thus, the performance of a single transputer, and of a network of transputers is compared with that of a variety of Von Neumann type processors. The effect of the choice of language on a system performance is also observed and assessed.

I. INTRODUCTION

For many years, scientists and engineers have demanded faster and better computers capable of performing complex mathematical tasks quickly. Recent technological developments have seen an explosion in the range of processors available off the shelf, ranging from the CISC based Intel 80386, to the RISC based SPARC range of processors.

The introduction of the Inmos transputer in 1985 has led to increased use of parallel and distributed processing, which has the potential to offer considerable performance improvements over traditional sequential processors. One notable work has been reported [1] comparing the transputer with other processors using a number of different algorithms. Though it presents a comparison between a limited range of processors, it does not concentrate on implementing a specific algorithm on a wide range of processors, or using a range of programming languages.

II. COMPUTERS IN ENGINEERING APPLICATIONS

II.1 CONVENTIONAL OR VON NEUMANN COMPUTERS

These computers are based upon a sequential processor, executing one instruction after another. There are essentially two variants of the Von Neumann Architecture. The first is the RISC processor, which is true to the original proposal, has a small instruction set and aims to be able to execute all instructions in one clock cycle. The CISC architecture, a derivative of the original Von Neumann proposal, has a larger instruction set at the expense of increased execution time. The aim of extending the instruction set is to make the processor easier to program using high level languages. However, the comparison between RISC and CISC processors is much more complex, and is adequately covered elsewhere [2].

The common Personal Computer comes into this class, as do the Sun Workstation range and the majority of Super-computers. For this paper, the following processors were used: 286, 486, Sun ELC, Sun IPC and Sun SPARCServer. To consider the power of super-computers, the Convex C220 at ULCC (University of London Computing Centre) was used.

II.2 Parallel Computers

Parallel machines constitute a number of on-board computers, each with its own CPU and communication links, and shared or local memory, which gives it the capability to operate as distributed structures. The ability to execute several instructions at once gives a considerable performance improvement over equivalent Von Neumann processors, taking into account the proper utilisation of a well scheduled distributed algorithm [3].

The transputer is a good example of a parallel processor, with 4 communications links, and on-chip memory. However, the protocols involved in the serial communications can severely degrade the system performance [6]. The transmitter must first ascertain whether the receiver is ready to receive, which requires two transmissions. Then the data itself must be sent, followed by two further transmissions to signal the end of communication. If a large array of data must be sent, the above protocol (i.e. sending one element at a time) can seriously affect the overall algorithm performance. However, sending the data in one large block is a better alternative, dramatically reducing the required communications time. Therefore, communications efficiency is one of the issues considered in this paper.

One major and common problem in using parallel processors is the occurrence of the dreaded deadlock. Time dependencies between processes can also cause problems for the uninitiated, but can be avoided by providing proper scheduling and distribution of the algorithm.

III. Benchmarks

The ideal benchmark for a processor is one that makes effective use of the processor and it's available resources, and reflects the typical workload of the processor. However, the benchmark should not test peripheral units such as mass storage, or display systems, as the performance of these can be variable.

There are a number of standard tests that fit these requirements, such as the Dhrystone (integer operations) & Whetstone tests (floating point operations), and Livermore Loops (loops with both integer and floating point maths) [4]. However, the Whetstone, Dhrystone, & Livermore Loops tests execute a pre-defined sequence of instructions, and bear no direct resemblance to any engineering problem. Also, the accuracy of the results are of no apparent consequence, and the test depends on the compiler/hardware combination, of which there are many variants. This latter point is one that cannot be eliminated easily, without resorting to assembly level programming, which is both tedious and impractical. Another drawback of the tests is that they only use small one-dimensional arrays, whereas many engineering problems employ large multidimensional arrays.

Therefore, it is required to find a simple, computationally intensive engineering related algorithm to use as a benchmark, such as the solution of a system of simultaneous linear equations using the Gaussian elimination [7]. The coefficients of the equations used contains

function such as sine() and cosine() to exploit the processor performance fully. In addition, the Gaussian elimination is a relatively simple convert to a parallel algorithm for the transputer.

The machines tested and reported on in this paper are as listed in table 1 where any hardware extras are indicated. This is not an exhaustive list of all the processors used, where others are included in the final report [5].

Machine	Processor	Typical Cost
IBM AT	IBM 80286 8MHz	£500
Viglen 486	Intel 80486DX 33MHz	£1,500
Sun ELC	40MHz SPARC RISC processor	£3,500
Sun IPC	25Mhz SPARC RISC processor	£4,000
Sun SPARCServer 2/670MP	2 SPARC processors + 2 Weitek Co-processors	£35,000
Convex C220		>> £35,000

Table 1. Sequential processors used for the benchmark tests

Both the Sun SPARCServer and the Convex are multi-user systems, and thus the results would be affected by system load. To allow for this, the test was run a number of times, at different times of day (and night), and an average taken. Although the SPARCServer is equipped with two processors, it must be emphasised that they do not operate in a distributed mode.

III.1 Results with Conventional Processors

Table 2 shows a selection of the results obtained by running the algorithm on the sequential processors listed above (the detailed results table runs to about 4 pages [5]). The MFLOP ratings shown in table 2 are derived from dividing the time taken for a given section of the algorithm divided by the number of real operations in the section. For example, if the instruction LET x=x*10 takes 1 second to execute, then it is 1 FLOP(=0.001 MFLOP).

The performance results of programmes written in FORTRAN and C are shown in figure 1 in terms of MFLOP ratings, where it can be seen that FORTRAN produces the best overall results compared to C. In addition, figure 2 shows the times taken by the processors for the individual sections of the benchmark algorithm.

Figure 2 shows that the Intel 486 processor performs better than the Sun IPC, whilst not as good as the Sun ELC. The Convex gives excellent performance with FORTRAN. However, without compiler vectorization, the Convex performance is rather poor, but with maximum vectorization its performance is much enhanced.

III.2 Parallel Results

This section includes the results obtained when running the benchmark algorithm on the transputer, showing the results obtained using the Inmos Ansi-C toolset. In addition to a sequential execution, a distributed Gaussian elimination algorithm was developed on two processors (see figure 3) and executed in both multitasking (on a single transputer) and parallel

(on two transputers) forms, with the results reported in table 3. However, the back-substitution part of the benchmark is kept in its sequential form. The utilisation of the two processors is approximately 80% and 70%, respectively. Nonetheless, although not employing an optimal scheduling scheme, the paralleled algorithm serves the purpose of this work, and presents a significant improvement over the sequential version.

Considering table 3, the two parallel transputers operate far more efficiently when only computations are assessed. Once communications are included, the performance decreases drastically, as discussed in section II.2. All the performance results obtained for the transputer are included in figures 1 and 2 for comparison.

Processor	Language	Compiler	Gaussian Elimination	Backsub-stitution	Average MFLOPS
IBM 286	Fortran	Prospero	434.02	2.08	7.01
i80486dx	Fortran	Salford	1.95	0.0199	1034.1268
Sun SPARCserver 2/670MP	Fortran	Sun	0.39	0.01	3471.9632
Sun IPC	Fortran	Sun	8.89	0.32	369.3375
Sun ELC	Fortran	Sun	0.61	0.01	2471.6656
Convex C220	Fortran	Convex	0.69	0.001835	5934.3243
IBM 286	Ansi-C	MicroSoft	542	5	3.51
i80486dx	Ansi-C	Gnu	4	0	525.0315
Sun SPARCserver 2/670MP	Ansi-C	Gnu	3	0	3055.626
Sun IPC	Ansi-C	Gnu	14	0	184.968
Sun ELC	Ansi-C	Gnu	3	0	1827.126
T800	Ansi-C	Inmos	6.73	0.035	417.544
Convex C220	Ansi-C	Convex	0.78	0	5064.95
i80486dx	Modula-2	Logitech	5.16	0.06	183.45833
i80486dx	Pascal	Sheffield	9.752	0.0448	290.46076

Table 2. Results for sequential machines (Seconds)

Processor	Language	Compiler	Execution mode	Gaussian Elimination	Back-substitution	Average MFLOPs
1 T800	Ansi-C	Inmos	Sequential	6.73	0.035	417.544
1 T800	Ansi-C	Inmos	Multitasking	8.00	0.053	296.438
2 T800	Ansi-C	Inmos	Parallel (no communications)	5.03	0.053	436.771
2 T800	Ansi-C	Inmos	Parallel (with communications)	9.11	0.053	294.902

Table 3. Parallel processor Results (Seconds)

Communications efficiency was assessed by sending the array element by element, and then repeating the exercise by sending the data in rows and as a single batch. The resulting times are shown in table 4 for both the multitasking and parallel forms.

Sending method	Multitasking on a single transputer	Parallel on two transputers
Element at a time (126x127 elements)	2.9087	3.0892
Row at a time (127 elements)	2.1475	2.7244
Whole matrix	1.588	1.803

Table 4. Communication times of a 126 by 127 matrix (Seconds)

IV. Discussions and Conclusions

Using the solution of a system of simultaneous linear equations as a benchmark algorithm represents a good performance index for engineering applications to assess and compare different processors. The algorithm satisfied the requirements of being computationally intensive, and demanding in terms of memory requirements. The performance results obtained from sequential tests shows that the Sun SPARCServer produced the best timing, followed by the Sun ELC. The Convex gives a far better performance with the clear drawback of a very high price. The performance of the transputer, as a single sequential processor, comes between the Sun IPC and the Sun ELC. Considering a distributed formulation, the inter-processors communications between the two processors burden the parallel performance to a large extent, and renders the two transputers to perform less than the IPC. However, once the computations are considered on their own, the two transputers performance holds its position between the IPC and ELC, although a better performance is expected on a larger network [7]. In addition, communications assessment showed a tremendous speed up by sending the data as one batch rather than individual elements.

Further tests are performed on variations of these computing machines, and employing Occam, parallel 3L FORTRAN and C versions of the software, and are included in the final report [5].

References

1. Stiles, G.S., 'How the Transputer stacks up to other Processors: A comparison of performance on Several Application Programs', *Mailshot*, SERC/DTI Transputer Initiative, November 1989.
2. Hwang, K. and Briggs, F.A., *Computer Architecture and Parallel Processing*, McGraw Hill, 1987.
3. Lewis, T.G. and H. El-Rewini, *Introduction to Parallel Computing*, Prentice-Hall, 1992.
4. Curnow, H.J. and Wichmann, B.A., 'A Synthetic Benchmark', *Computer Journal*, Volume 19, No. 1, February 1976, PP.43-49.
5. Cornish, I.J., *Transputer Benchmarks and Performance Comparisons with other Computing Machines*, Final year thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, 1993.
6. Inmos Ltd, *Transputer Reference Manual*, Prentice-Hall International, 1988.
7. Roberts, Y. *The impact of Vector and Parallel Architectures on the Gaussian Elimination algorithm*, Manchester University Press, 1990.

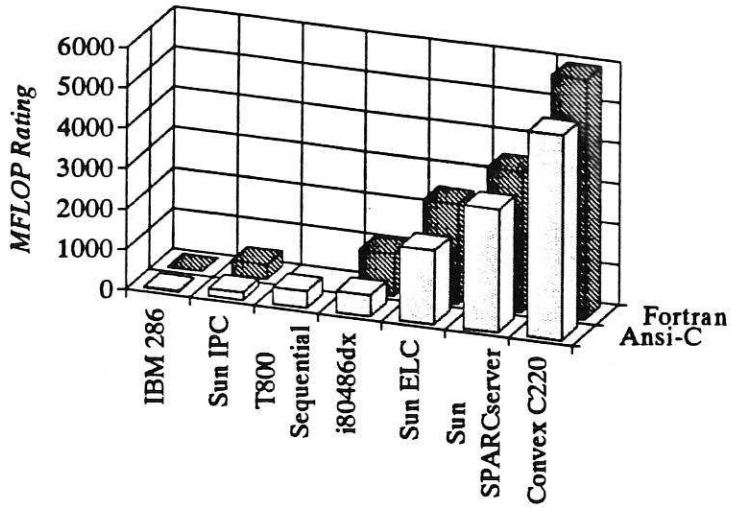

Figure 1: MFLOP ratings for selected C & FORTRAN results

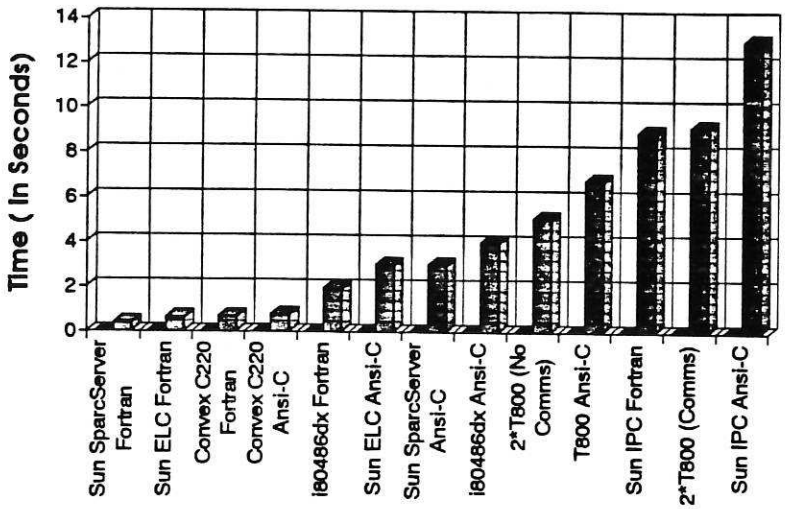

Figure 2: Execution times on processors (seconds)

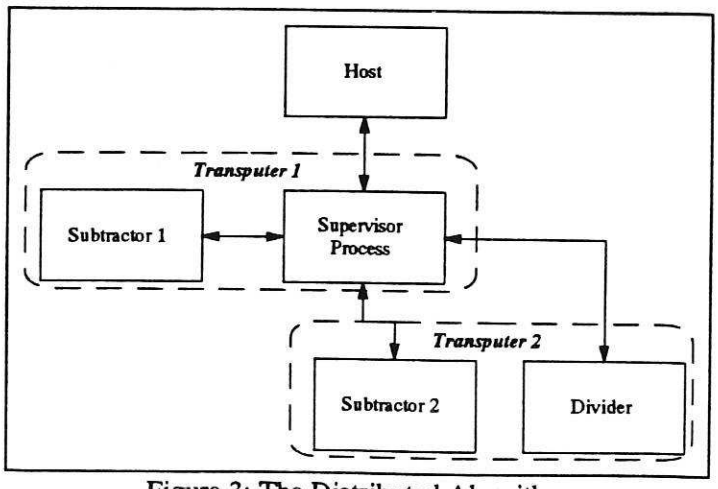

Figure 3: The Distributed Algorithm