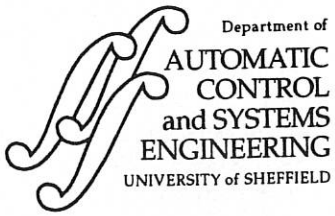
Monograph:
Tokhi, M.O. and Ramos-Hernandez, D.N. (1998) Performance Evaluation Metrics and Load-Balanced Task to Processor Allocation in Parallel Architectures. UNSPECIFIED. ACSE Research Report 735 . Department of Automatic Control and Systems Engineering



PERFORMANCE EVALUATION METRICS AND LOAD-BALANCED TASK TO PROCESSOR ALLOCATION IN PARALLEL ARCHITECTURES

M. O. Tokhi and D. N. Ramos-Hernandez

Department of Automatic Control and Systems Engineering,
The University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK.

Tel: + 44 (0)114 222 5617.
Fax: + 44 (0)114 273 1729.
E-mail: o.tokhi@sheffield.ac.uk.

Research Report No. 735

December 1998

Abstract

This paper presents an investigation into the development of performance evaluation metrics for sequential and parallel architectures. Speedup and efficiency are defined using the concept of virtual processor. These are utilised to obtain the best task allocation to processors in parallel architectures achieving maximum efficiency and speedup. The performance metrics developed are general and applicable to both heterogeneous and homogeneous architectures and ensure that capabilities of the processors are exploited by maximising the efficiency of the architecture. The proposed concepts are validated experimentally using several algorithms and architectures including digital signal processing and message-passing systems.

Keywords: Efficiency, heterogeneous architectures, homogeneous architectures, parallel processing, speedup, task allocation.

CONTENTS

LIST OF FIGURES

1 Introduction

The performance demands of modern signal processing and control applications require the employment of complex algorithms with varying computational requirements. Various types of processing elements (PEs) are designed to fulfil different computational requirements. Digital signal processing (DSP) devices are designed in hardware to perform concurrent add and multiply instructions and execute irregular algorithms efficiently, typically finite-impulse response (FIR) and infinite-impulse response (IIR) filter algorithms. The Intel i860 vector processor, for instance, has been designed for high performance floating point computation and is able to efficiently process regular algorithms involving matrix manipulations. Purpose built PEs are not enough to bridge the ever-increasing software/hardware gap. Alternative strategies where multi-processor based systems are employed, utilising high performance reduced instruction set computer (RISC) processors, DSP devices, transputers and parallel processing (PP) techniques, could provide suitable methodologies.

For PP with widely different architectures and different PEs, performance measurements such as million instructions per second (MIPS), million operations per second (MOPS) and million floating point operations per second (MFLOPS) of the PEs are meaningless. Of more importance is to rate the performance of each architecture with its PEs on the type of program likely to be encountered in a typical application. While the different architectures and their different clock rates, memory cycle times of the PEs, inter-processor communication speed, optimisation facility and compiler performance are influential factors, they confuse the issue of attempting to rate an architecture. This is an inherent difficulty in selecting a parallel architecture, for better performance, for algorithms in signal processing and control system development applications. The ideal performance of a parallel architecture demands a perfect match between the capability of the architecture and the program behaviour (Tokhi and Hossain, 1995a,b). Capability of the architecture can be enhanced with better hardware technology, innovative architectural features and efficient resources management. In contrast, program behaviour is difficult to predict due to its heavy dependence on application and run-time conditions. Moreover, there are many other factors that influence program behaviour. These include algorithm design, partitioning and

mapping of the algorithm, inter-processor communication, data structures, language efficiency, programmer skill, and compiler technology (Tokhi *et al.*, 1997b).

A commonly used measure of performance of a processor in an application is speedup. This is defined as the ratio of execution time of the processor in implementing the application algorithm relative to a reference time or execution time of a reference processor (Tokhi *et al.*, 1997a). The speedup thus defined provides a relative performance measure of a processor for fixed load (task size) and thus can be referred to as fixed-load speedup. This can also be used to obtain a comparative performance measure of a processor in an application with fixed task sizes under different processing conditions, for example, with and without code optimisation.

Speedup is also one of the most commonly used metrics for parallel processing. In this context, there are three known speedup performance models: fixed-size (fixed-load) speedup, fixed-time speedup and memory-bounded speedup (Sun and Ni, 1993; Sun and Rover, 1994). Fixed-size speedup fixes the problem size (load) and emphasises how fast a problem can be solved. Fixed-time speedup argues that parallel computers are designed for otherwise intractably large problems. It fixes the execution time and emphasises how much more work can be done with parallel processing within the same time. Memory-bounded speedup assumes that the memory capacity, as a physical limitation of the machine, is the primary constraint on large problem sizes. It allows memory capacity to increase linearly with the number of processors. Both fixed-time and memory-bounded speedups are forms of scaled speedups. The term scaled-speedup has been used for memory-bounded speedup by many authors (Gustafson, 1988; Nussbaum and Agrawal, 1991).

When speed is the goal, the power to solve problems of some magnitude in a reasonably short period of time is sought. Speed is a quantity that ideally would increase linearly with system size. Based on this reasoning, the isospeed approach, described by the average unit speed as the achieved speed of a given computing system divided by the number of processors N, has previously been proposed (Sun and Rover, 1994). This provides a quantitative measure of describing the behaviour of a parallel algorithm-machine combination as sizes are varied.

Another useful measure in evaluating the performance of a parallel system of N processors is efficiency. Efficiency can be interpreted as providing an indication of the average utilisation of the 'N' processors, expressed as a percentage. Furthermore, this measure allows a uniform comparison of the various speedups obtained from systems containing different number of processors. It has also been illustrated that the value of efficiency is directly related to the granularity of the system.

Although, speedup and efficiency and their variants have widely been discussed in relation to homogeneous parallel architectures. Not much has been reported on such performance measures for heterogeneous parallel architectures.

Due to substantial variation in computing capabilities of the PEs, the traditional parallel performance metrics of homogeneous architectures are not suitable for heterogeneous architectures in their current form. Note, for example, that speed up and efficiency provide measure of performance of parallel computation relative sequential computation on a single processor node. In this manner, the processing node is used as reference node. In a heterogeneous architecture, such a reference node, representing the characteristics of all the PEs, is not readily apparent. In this investigation such a reference node is identified by proposing the concept of virtual processor. Moreover, it is argued in this context that a homogeneous architecture can be considered as a sub-class of heterogeneous architectures. In this manner, the performance metrics developed for heterogeneous architectures should be general enough to cover both classes of architectures. The metrics developed are shown to satisfy this requirement.

Attempts have previously been made at proposing speedup of a heterogeneous architecture as the ratio of minimum sequential execution time among the PEs over parallel execution time of the architecture (Yan *et al.*, 1996; Zhang and Yan, 1995). In this manner, the best PE in the architecture is utilised as the reference node and efficiency of the architecture is defined accordingly. Although, it has been shown that the definitions under a specific situation transform to those of a homogeneous architecture, such transformation does not in general hold. Moreover, the concept does not fully exploit the capabilities of all the PEs in the architecture for purposes of task to processor mapping and scheduling.

Instead, it relies on the performance of the best PE. The concept proposed in this paper ensures that the capabilities of the PEs are exploited by maximising the efficiency of the architecture. The organisation of the paper is as follows.

Section 2 describes the algorithms implemented to model and measure the performance metrics and the hardware used for implementation. The proposed sequential and parallel performance metrics are introduced in Section 3. Section 4 describes the task allocation strategy for parallel architectures. Section 5 presents implementations and results and finally, the paper is concluded in Section 6.

2 Algorithms and architectures

To evaluate the performance of parallel architectures in signal processing and control applications, two algorithms are considered in this investigation. The first algorithm is a cantilever beam simulation algorithm. This is of regular nature and is based on matrix multiplication and addition. The second algorithm is a basic linear algebraic operation referred as DOT algorithm. These algorithms are implemented on a number of architectures incorporating the Intel i860 RISC processor (i860), the Texas Instruments TMS320C40 (C40) DSP devices and INMOS T805 (T8) transputers. These are briefly described below.

2.1 The beam simulation algorithm

Consider a cantilever beam system with a force $U(x,t)$ applied at a distance x from its fixed end at time t. This produces a deflection $y(x,t)$ of the beam from its stationary position at the point where the force has been applied. The dynamic equation describing the system is (Tokhi and Hossain, 1994)

$$\mu^2 \frac{\partial^4 y(x,t)}{\partial x^4} + \frac{\partial^2 y(x,t)}{\partial t^2} = \frac{1}{m} U(x,t) \qquad (1)$$

where μ is a beam constant, m is the mass of the beam.

Using a finite difference discretisation of the dynamic equation of the beam in time and distance yields (Tokhi and Hossain, 1994)

$$Y_{k+1} = -Y_{k-1} - \lambda^2 S Y_k + \frac{(\Delta t)^2}{m} U(x,t) \qquad (2)$$

where S is a pentadiagonal matrix, entries of which depend on the physical properties and boundary conditions of the beam, Y_i $(i = k+1, k, k-1)$ represents the deflection at sections (grid-points) 1,...,n of the beam at time step i, Δt and Δx are increments along time and distance coordinates respectively, and $\lambda^2 = (\Delta t)^2 (\Delta x)^{-4} \mu^2$.

Equation (2) comprises the beam simulation algorithm which can easily be implemented on a digital processor.

2.2 *The DOT algorithm*

The DOT algorithm is a basic linear algebraic operation, which incorporates floating-point add and multiply operations (Sun and Gustafson, 1991). This is given as

$$y(i) = a + b(i) \times c(i) \qquad (3)$$

where a, b and c represent real numbers.

2.3 *The hardware architectures*

The topology of the heterogeneous architecture used to carry out the experimental investigations in this work is shown in Figure 1. This comprises a Transtech TMB08 motherboard with 10 TRAMs possessing two T8s. The root T8 incorporates 2 Mbytes local memory and is connected to the Host computer (SUN SPARC), via its link 3 and to the Transtech TMB16 motherboard that contains the i860 TRAM. The i860 TRAM has 16 Mbytes shared memory and a T805 with 4 Mbytes of local memory. The second transputer on the TMB08 motherboard is connected to the root transputer via its link 1 and to a sub-network of three C40s each with 3 Mbytes DRAM and 1 Mbyte SRAM.

The T805 transputer is a 32 bit CMOS microcomputer with a 64 bit floating point unit and graphics support. It is a general-purpose medium-grained parallel processor with 25 MHz clock speed, yielding up to 20 MIPS performance and is capable of 4.3 MFLOPS. This has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface

and four standard INMOS communications links. The links operate at speeds of 20 Mbits/sec, achieving data rates of up to 1.7 Mbytes/sec uni-directionally or 2.3 Mbytes bi-directionally. The T805 can directly access a linear address space of 4 Gbytes and a configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems (Inmos, 1989).

The C40 is a 32-bit DSP processor with 40 MHz clock speed, 8 Kbytes on-chip RAM, and 512 bytes on-chip instruction cache. It is capable of 275 MOPS and 40 MFLOPS. This processor has six communication ports for high speed inter-processor communication (20 Mbytes/sec asynchronous transfer rate at each port for maximum data throughput), six-channel DMA coprocessor for concurrent I/O and CPU operation, two identical external data and address buses supporting shared memory systems and high data rate (single-cycle transfers), on-chip program cache and dual-access/single-cycle RAM for increased memory access performance and separate internal program data (Texas Instruments, 1991).

The i860 is a 64-bit RISC processor with 40 MHz clock speed, a peak integer performance of 40 MIPS, 8 Kbytes data cache and 4 Kbytes instruction cache. It is capable of 80 MFLOPS. This processor has separate integer, floating-point, graphics, adder, multiplier and memory-management units. Furthermore, both the integer unit and the floating-point control unit can execute concurrently. In this case, the i860 is also a superscalar RISC processor capable of executing two instructions, one integer and one floating-point standard, operating with single-precision (32-bit) and double-precision (64-bit) operands. The graphics unit supports three-dimensional drawing in a graphics frame buffer, with colour intensity, shading, and hidden surface elimination (Hwang, 1993).

3 Performance metrics

In this section general measures of speedup for sequential and parallel architectures are proposed and used with a proposed task allocation strategy among parallel PEs to achieve maximum efficiency.

3.1 Sequential processing

It has previously been reported that the performance of a processor, as execution time, in implementing an application algorithm generally evolves linearly with the task size (Tokhi *et al.*, 1996; Tokhi *et al.*, 1997a). With some processors, however, anomalies in the form of change of gradient (slope) of execution time to task size are observed. These are mainly due to run-time memory management conditions of the processor where, up to a certain task size the processor may find the available cache sufficient, but beyond this it may require to access lower level memory. Despite this the variation in the slope is relatively small and the execution time to task size relationship can be considered as linear. This means that a quantitative measure of performance of a processor in an application can adequately be given by the average ratio of task size to execution time or the average speed. Alternatively, the performance of the processor can be measured as the average ratio of execution time per unit task size, or the average (execution time) gradient. In this manner, a generalised performance measure of a processor relative to another in an application can be proposed.

Let the average speeds with two processors p_1 and p_2 in an application, over a range of task sizes, be denoted by V_1 and V_2 respectively. The generalised sequential (execution time) speedup $S_{1/2}$ of p_1 relative to p_2 in implementing the application algorithm can thus be defined as

$$S_{1/2} = \frac{V_1}{V_2} \tag{4}$$

Alternatively, if the corresponding average gradients with p_1 and p_2 for the application, over a range of task sizes, are given by G_1 and G_2 respectively, $S_{1/2}$ can be expressed as

$$S_{1/2} = \frac{G_2}{G_1} \tag{5}$$

The concept of generalised sequential speedup described above can also be utilised to obtain a comparative performance evaluation of a processor for an application under various processing conditions, for example with and without code optimisation.

The concept of speed, assumed to be constant for a processor in an application, has previously been utilised to derive an expression for the generalised speedup of a parallel (homogeneous) architecture as the ratio of parallel speed (of the parallel architecture) over sequential speed (of a single processor) (Sun and Gustafson, 1991; Sun and Rover, 1994). The generalised speedup introduced above, however, reflects on the relative performance of two uni-processor architectures in an application and of the same processor under two different processing conditions.

3.2 Parallel processing

Consider a heterogeneous parallel architecture of N processors. To define speedup and efficiency of the architecture, assume a virtual processor is constructed that would achieve a performance, in terms of average speed, equivalent to the average performance of the N processors. Let the performance characteristics of processor i $(i = 1,\ldots,N)$ over task increments of ΔW be given by

$$\Delta W = V_i \Delta T_i \tag{6}$$

where ΔT_i and V_i represent the execution time increment and average speed of the processor. Thus, the speed V_v and execution time increment ΔT_v of the virtual processor executing the task increment ΔW are given as

$$V_v = \frac{\Delta W}{\Delta T_v} = \frac{1}{N}\sum_{i=1}^{N} V_i = \frac{1}{N}\sum_{i=1}^{N} \frac{\Delta W}{\Delta T_i} = \frac{\Delta W}{N}\sum_{i=1}^{N} \frac{1}{\Delta T_i} \tag{7}$$

and

$$\Delta T_v = N\left(\sum_{i=1}^{N} \frac{1}{\Delta T_i}\right)^{-1} \tag{8}$$

Thus, the fixed-load increment parallel speedup S_f and generalised parallel speedup S_g of the parallel architecture, over a task increment of ΔW, can be defined as

$$S_f = \frac{\Delta T_v}{\Delta T_p}$$
$$S_g = \frac{V_p}{V_v} \tag{9}$$

where ΔT_p and V_p represent the execution time increment and average speed of the parallel system. In this manner, the (fixed-load) efficiency E_f and generalised efficiency of the parallel architecture can be defined as

$$E_f = \frac{S_f}{N} \times 100\%$$
$$E_g = \frac{S_g}{N} \times 100\% \tag{10}$$

Note in the above that the concepts of parallel speedup and efficiency defined for heterogeneous architectures are consistent with the corresponding definitions for homogeneous architectures. Thus, these can be referred to as the general definitions of speedup and efficiency of parallel architectures.

4 Task to processor allocation in parallel architectures

The concept of generalised sequential speedup can be utilised as a guide to allocation of tasks to processors in parallel architectures so as to achieve maximum efficiency and maximum (parallel) speedup. Let the generalised sequential speedup of processor i (in a parallel architecture) to the virtual processor be $S_{i/v}$;

$$S_{i/v} = \frac{V_i}{V_v} \; ; \quad i = 1,\ldots,N \tag{11}$$

Using the processor characterisations of equations (6) and (7) for processor i and the virtual processor, equation (11) can alternatively be expressed in terms of fixed-load execution time increments as

$$S_{i/v} = \frac{\Delta T_v}{\Delta T_i} \; ; \quad i = 1,\ldots,N \tag{12}$$

Thus, to allow 100% utilisation of the processors in the architecture the task increments ΔW_i allocated to processors $i = 1,\ldots,N$ should be so that the execution time increment of the parallel architecture in implementing the task increment ΔW is given by

$$\Delta T_p = \Delta T_i = \frac{\Delta W_i}{V_i} = \frac{\Delta T_v}{N} = \frac{1}{N}\frac{\Delta W}{V_v} \quad ; \quad i = 1,\ldots,N \tag{13}$$

or, using equation (11),

$$\Delta W_i = \frac{V_i}{V_v}\frac{\Delta W}{N} = S_{i/v}\frac{\Delta W}{N} \quad ; \quad i = 1,\ldots,N \tag{14}$$

It follows from equation (13) that, with the distribution of load among the processors according to equation (14) the parallel architecture is characterised by

$$\Delta W = (NV_v)\Delta T_p = V_p \Delta T_p \tag{15}$$

having an average speed of

$$V_p = NV_v \tag{16}$$

Thus, with the distribution of load among the processors according to equation (14), the speedup and efficiency achieved with N processors are N and 100% respectively. These are the ideal speedup and efficiency. In practice, however, due to communication overheads and run-time conditions the speedup and efficiency of the parallel architecture will be less than these values.

Note in the above that, in developing the performance metrics for a heterogeneous parallel architecture of N processors, the architecture is conceptually transformed into an equivalent homogeneous architecture incorporating N identical virtual processors. This is achieved by the task allocation among the processors according to their computing capabilities to achieve maximum efficiency. For a homogeneous parallel architecture the virtual processor is equivalent to a single PE in the architecture.

5 Implementations and results

In implementing the beam simulation algorithm a cantilever beam of length $L = 0.635$ m and mass $m = 0.037$ kg was considered. The algorithm granularity was achieved by increasing the number of beam segments from 5 to 40, in increments of 5. The execution times of the processors were obtained over 20000 iterations in each case.

In the case of the DOT algorithm the number of data points was varied from 1000 to 10000, in increments of 1000. For simplicity purposes, the vectors $b(i)$ and $c(i)$ were fixed to 4.0 and 6.5 respectively.

Figure 2 shows the execution times of the C40, T8 and the integrated C40&T8 architectures in implementing the beam simulation algorithm. The characteristics of the virtual processor and the corresponding theoretical C40&T8 heterogeneous architecture are also shown in Figure 2. It is noted that among the uni-processors, the C40 is considerably faster that the T8. The combination, thus, results in a virtual processor with characteristics closer to that of the C40. This implies that, for the C40&T8 to achieve maximum efficiency, a large proportion of the task is to be allocated to the C40. It is noted that the actual C40&T8 has performed slower than the corresponding theoretical (100% efficient) model. This is due to the communication overhead between the processors.

In implementing the DOT algorithm, the heterogeneous architecture of T8 and C40 was utilised. Figure 3 shows the execution times achieved with the architectures in implementing the algorithm. It is noted that the performance of the actual C40&T8 is close to its corresponding theoretical model. The generalised speedup and efficiency achieved with the C40&T8 in this case are 2 and 100% respectively.

Another heterogeneous configuration was used to implement the beam simulation algorithm. This consisted of one T8 and one i860. Figure 4 shows the execution times corresponding to this implementation. It is noted that the i860 is extremely faster than the T8. For this reason, all the task was allocated to the i860, with the exception of the last two values (35 and 40 segments) where one segment was allocated to the T8. The performance of the actual T8&i860 is similar to the uni-processor implementation with the i860.

However, for the last two values inter-processor communication has increased the execution time for the actual implementation.

Figure 5 shows the execution times of the C40, i860 and the actual, and theoretical execution times of the C40&i860 heterogeneous architecture in implementing the beam simulation algorithm. It is noted that the actual implementation is faster than the virtual processor and faster than the i860. It is also noted that the actual implementation is very close to the virtual parallel machine.

Comparing the three implementations of the beam simulation algorithm, it is noted that with the T8&C40, the execution times of the actual implementation stay very close to the virtual processor. Task allocation was computed according to the theoretical model. However, communication overheads are evident with the T8&i860 implementation where no task is allocated to the T8 except for the case of 35 and 40 segments. This resulted in the same execution times as the i860 processor, except for the 35 and 40 segment steps where the execution time has increased. The results of these two implementations show that due to the disparity in capabilities of the processors, communication overhead becomes a dominant factor in the implementation. The i860&C40 implementation shows that a better task allocation is obtained between both processors and communication overheads are minimum. In this manner the best performance for this algorithm, among the architectures, is obtained with the combination of the C40&i860.

6 Conclusion

An investigation into the performance evaluation of sequential and parallel computing has been carried out. Performance metrics, on the basis of maximum efficiency, have been proposed for parallel architectures. These apply to both homogeneous and heterogeneous architectures and are consistent with those of traditional architectures. These have been verified through implementation of two typical algorithms on uni- and multi-processor architectures.

Based on the proposed concept of speed up a task allocation strategy for heterogeneous architectures has been developed. It has been demonstrated that with such a strategy, maximum efficiency with a heterogeneous architecture can be achieved. Further investigations will consider inclusion of communication overheads and run-time conditions.

7 Acknowledgements

D. N. Ramos-Hernandez acknowledges the financial support of CONACYT-MEXICO.

8 References

GUSTAFSON, J. L. (1988). Re-evaluating Amdahl's law, *Communications of ACM*, **31**, (5), pp. 532-533.

HWANG, K. (1993). *Advanced computer architecture: Parallelism, scalability, programmability*. McGraw-Hill.

INMOS. (1989). *Transputer databook*. Second Edition. Redwood Burn Ltd, Trowbridge.

NUSSBAUM, D. and AGRAWAL, A. (1991). Scalability of parallel machines, *Communications of the ACM*, **34**, (3), pp. 57-61.

SUN, X. H. and GUSTAFSON, J. (1991). Toward a better parallel performance metric, *Parallel Computing*, **17**, (10), pp 1093-1109.

SUN, X. H. and NI, L. (1993). Scalable problems and memory-bounded speedup, *Journal of Parallel and Distributed Computing*, **19**, pp. 27-37.

SUN, X. H. and ROVER, D. T. (1994). Scalability of parallel algorithm-machine combinations, *IEEE Transactions on Parallel and Distributed Systems*, **5**, (6), pp 599-613.

TEXAS INSTRUMENTS. (1991). *TMS320C4x User's Guide*, Texas Instruments.

TOKHI, M. O. and HOSSAIN, M. A. (1994). Self-tuning active vibration control in flexible beam structures, *Proceedings of IMechE-I: Journal of Systems and Control Engineering*, **208**, (14), pp. 263-277.

TOKHI, M. O. and HOSSAIN, M. A. (1995a). CISC, RISC and DSP processors in real-time signal processing and control, *Microprocessors and Microsystems*, **19**, (5), pp. 291-300.

TOKHI, M. O. and HOSSAIN, M. A. (1995b). Homogeneous and heterogeneous parallel architectures in real-time signal processing and control, *Control Engineering Practice*, **3**, (12), pp. 1675-1686.

TOKHI, M. O., CHAMBERS, C. and HOSSAIN, M. A. (1996). Performance evolution with DSP and transputer based systems in real-time signal processing and control applications, *Proceedings of UKACC International Conference on Control-96*, Exeter, 2-5 September 1996, **1**, pp. 371-375.

TOKHI, M. O. , HOSSAIN, M. A. and CHAMBERS, C. (1997a). Performance evaluation of DSP and transputer based systems in sequential real-time applications, *Microprocessors and Microsystems*, **21**, pp. 237-248.

TOKHI, M. O., HOSSAIN, M. A., BAXTER, M. J. and FLEMING, P. J. (1997b). Performance evaluation issues in real-time parallel signal processing and control, *Journal of Parallel and Distributed Computing*, **42**, (1), pp. 67-74.

YAN, Y., ZHANG, X. and SONG, Y. (1996). An effective and practical performance prediction model for parallel computing on nondedicated heterogeneous networks of workstations, *Journal of Parallel and Distributed Computing*, **38**, (1), pp. 63-80.

ZHANG, X. and YAN, Y. (1995). Modeling and characterizing parallel computing performance on heterogeneous networks of workstations, *Proceedings of Seventh IEEE Symposium on Parallel and Distributed Processing*, San Antonio, Texas, 25-28 October 1995, pp. 25-34.

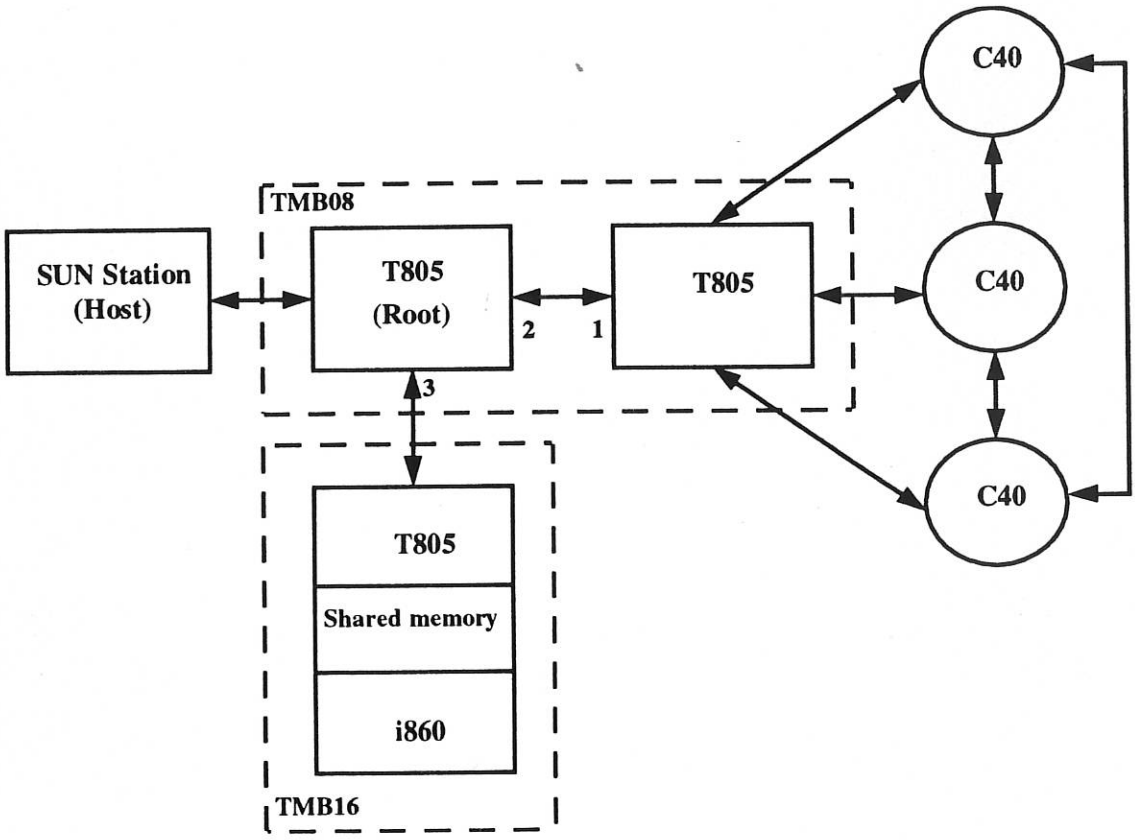


Figure 1: Topology of the heterogeneous architecture.

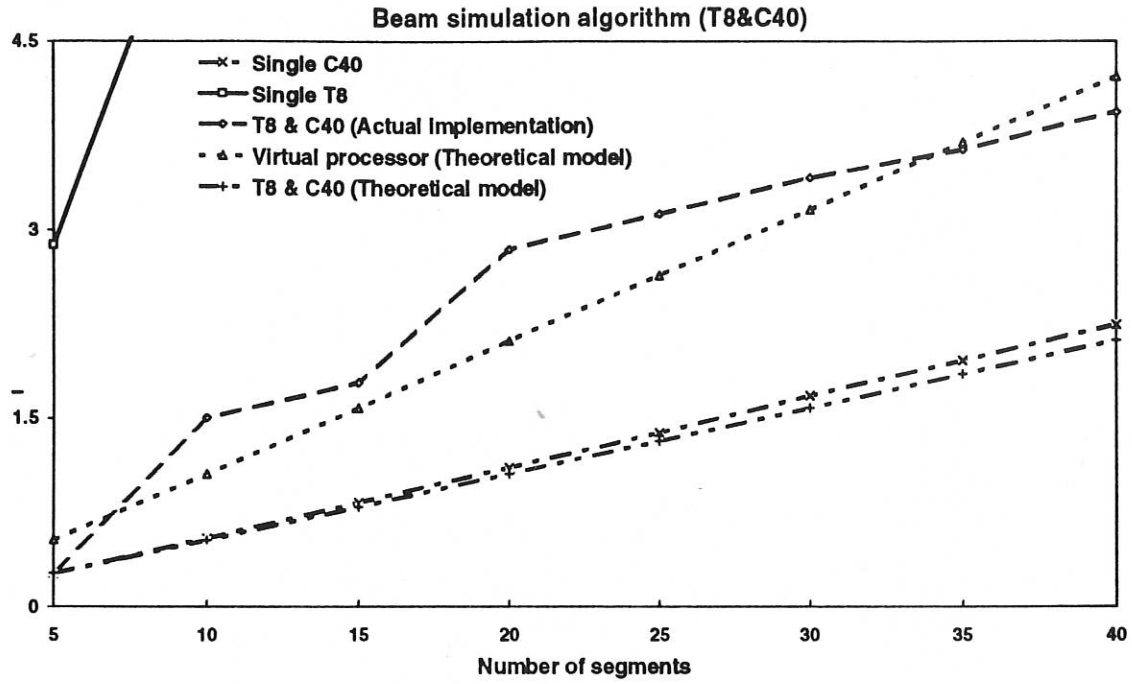


Figure 2: Execution times of the C40-T8 architectures in implementing the beam simulation algorithm.

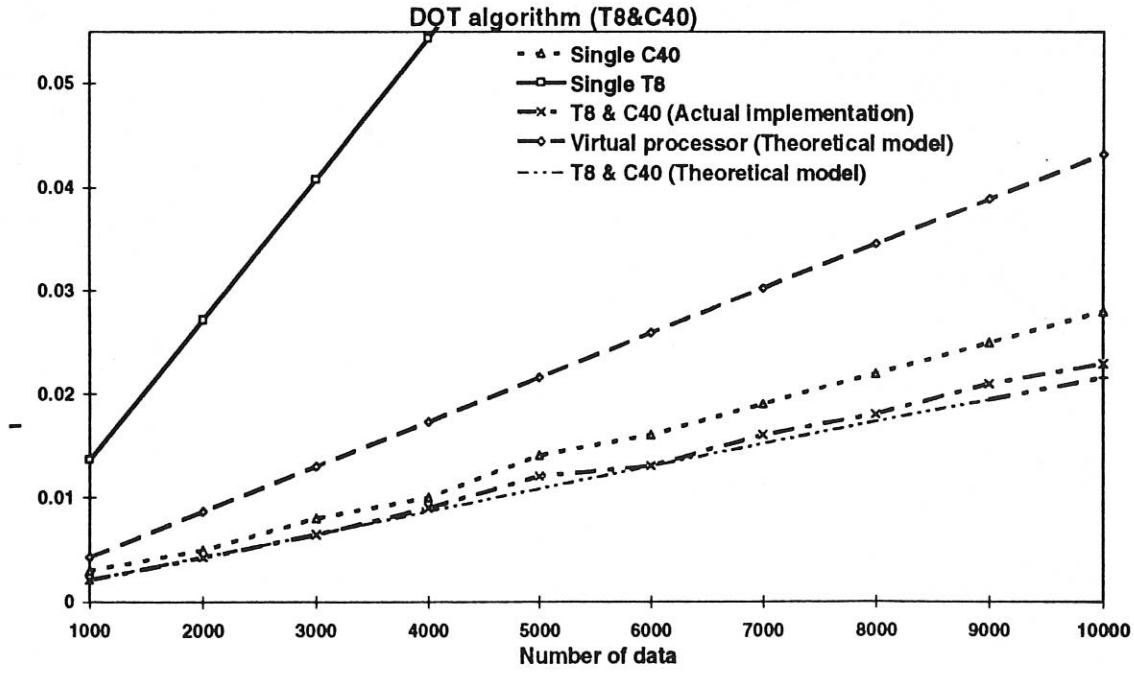


Figure 3: Execution times of the C40-T8 architectures in implementing the DOT algorithm.

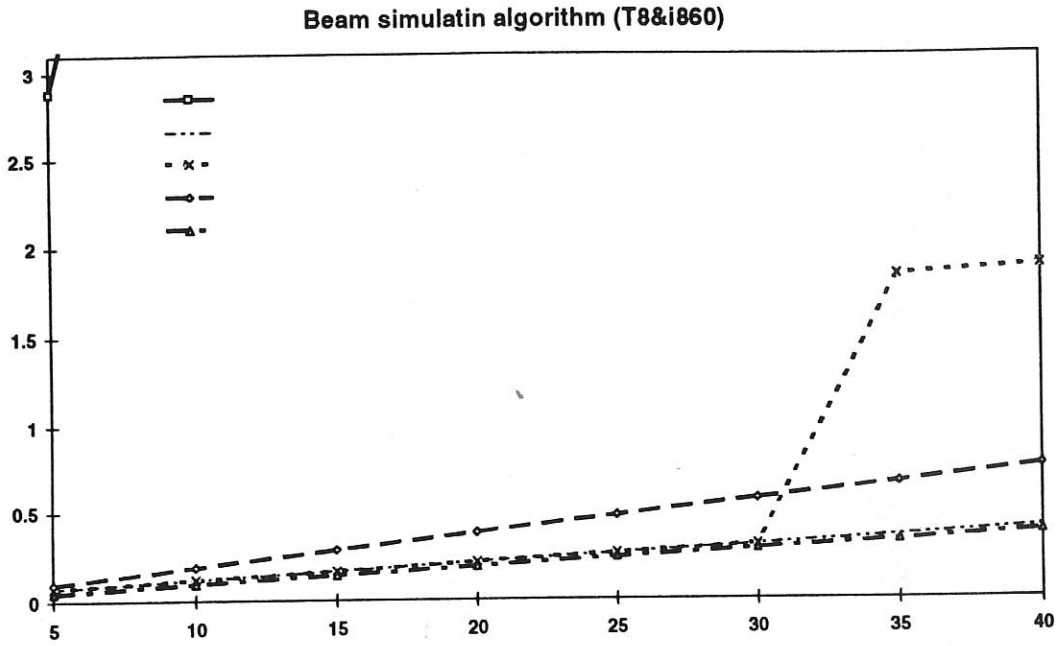


Figure 4: Execution times of the i860-T8 architectures in implementing the beam simulation algorithm.

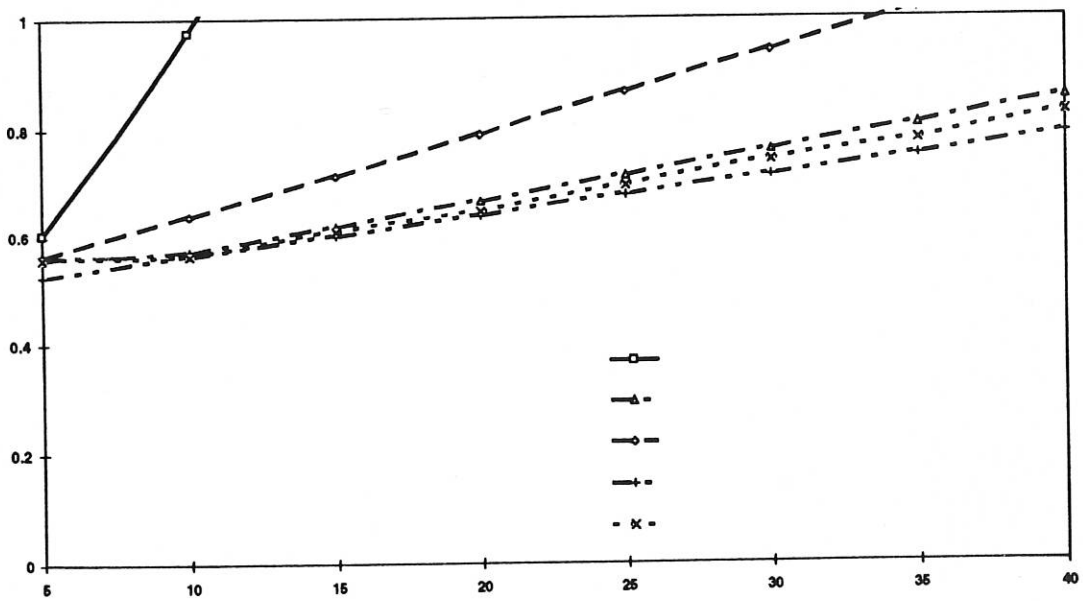


Figure 5: Execution times of the i860-C40 architectures in implementing the beam simulation algorithm.