Cost-Performance Analysis of Heterogeneity in Supercomputer Architectures*

Daniel Menascé
Departamento de Informática
Pontifícia Universidade Católica
22453, Rio de Janeiro, Brazil
Menasce@BRLNCC.BITNET

Virgílio Almeida
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
30161, Belo Horizonte, Brazil
Virgilio@BRLNCC.BITNET

Abstract

Heterogeneity has appeared as a cost-effective approach to design high performance computers. This paper analyzes cost-performance of heterogeneity in supercomputer architectures. Queueing models are used to study performance of homogeneous and heterogeneous supercomputer models. Grosch's Law, which states that computer performance increases as the square of its cost, is used to analyze cost aspects of the models. The results of this paper show that heterogeneity in supercomputer architectures is a quite promising design approach that deserves further investigation.

1 Introduction

Advances in VLSI technology have caused a dramatic change in cost-performance trade-offs for designing high performance computer systems. Performance has been increased by architectural innovations and progress in semi-conductor technology. Currently, computer systems consisting of up to tens of hundreds of relatively inexpensive microprocessors can be built. A number of machine architectures with a large number of processors have been implemented in an attempt to provide supercomputer power at a fraction of supercomputer cost (e.g., Intel iPSC, Ncube, Connection Machine, etc [17]). On the other hand, a supercomputer such as the NEC SX2 [17], consisting of a single processor is considered one of the most powerful machines available today.

Both design approaches, parallel and sequential processing, present some limitations. Powerful sequential processors are very expensive and their performance cannot be improved much further because they are reaching the ultimate limit, represented by the speed of light. Parallel systems circumvent this limitation by harnessing many relatively inexpensive VLSI processors together. More important, there is no foreseeable limit to the computing power that may be achieved through parallel processing. However, the sequential fraction of a computation places a rather significant constraint on the effectiveness with which any particular algorithm can make use of a large number of processors [9, 7, 10].

This paper analyzes heterogeneity in supercomputer architecture as a cost-effective approach to combine the best features of sequential and parallel processing: speed of sequential computation and unlimited growth of computing power in parallel processing. In parallel processing, heterogeneity concerns the use of different processors, that are dedicated to specific tasks and cooperate closely on the same job. Performance of heterogeneous parallel architectures is studied through the use of models based on queueing theory. This paper also analyzes cost aspects of heterogeneous architectures. In evaluating cost of computer systems, we refer to Grosch's Law [4, 1, 10], which states that computer performance increases as the square of its cost. This Law has been verified extensively with empirical data over generations of computers. Ein-Dor [1] points out that Grosch's Law is still valid if families of computers, such as supercomputers, mainframes, minicomputers and microcomputers are considered. There are economies of scale within any given computer category, but there are diseconomies in transition from one family to another. This paper uses Grosch's Law in order to construct heterogeneous models with the same cost of homogeneous parallel ones.

Ercegovac [3] discusses some approaches to heterogeneous architectures in order to achieve cost-effective performance and programmability. Wal-

*This research was partially supported by a grant from IBM Brasil

lach [18] also sugests that in the future high performance parallel systems will be built with heterogeneous processors as contrasted to homogeneous ones. However, no quantitative results are presented in those references [3, 18]. Queueing network models have been successfully used to investigate and compare hypothetical architectures. For instance, Heidelberg [6] uses queueing network models to study the performance of a hypothetical multimicroprocessor back-end database machine relative to that of a mainframe database system. Menasce and Almeida [2] also used analytic models to propose variations in supercomputer architectures. We are not aware of any paper that studies the cost-performance tradeoffs of heterogeneity in supercomputer architectures using analytic models.

2 Homogeneous and Heterogeneous Parallel Processing Paradigms

Rather than studying specific machines, this paper takes a more abstract view, and explores cost-performance trade-offs of heterogeneous versus homogeneous supercomputer architectures. In order to carry the analysis, two basic models of processing, which underlie the whole spectrum of high performance systems, are defined. The two models of processing that constitute the basic computation paradigms are the following: centralized and parallel processing. In the first model, a single processor performs the sequential execution of all tasks of a job. In a parallel processing system, a number of processors is organized in such a way that they cooperatively execute a single job, where each processor is assigned to execute a task. The models are analyzed in a context of a specific application domain, namely high volume on-line transaction processing. The Input/Output subsystem will not be considered here, since in this study we are basically interested in contrasting processor power versus cost in various architectures.

The centralized model assumes the existence of a central processing facility, consisting of a single processor, to which jobs are submitted for execution. Jobs that arrive and find the central facility busy have to wait in a queue. Machines such as the Hitachi S810/20, Fujitsu VP-200, and IBM 3090/180-VF [14] to name a few fall into this category.

Parallel processing architectures are divided into two classes: homogeneous and heterogeneous systems. The homogeneous parallel processing paradigm assumes the existence of a set of P ($P > 1$) identical processors which collectively process one job at a time. Jobs arrive from an infinite population source; those which find the set of P processors busy working on a given job will have to wait in a queue. It is assumed that a fraction F_p of the instructions of a job may be executed in parallel by the P processors (each one will execute $\frac{1}{P}$ of the total number of instructions that can be executed in parallel), and, a fraction F_s (equal to $1 - F_p$) of the instructions of the job must be executed sequentially by any one of the P processors. We will assume that the parallelizable part of the job can make simultaneous use of all P processors (i.e. logical parallelism is at least equal to the physical parallelism). Different portions of the same job may be executing in parallel in a cooperative fashion implying in an overhead associated with the necessary synchronization, communication and any other contention for shared resources (e.g. memory contention delays in shared memory multiprocessors). Amdahls's Law has demonstrated that the serial fraction of processing dominates the execution time for any large parallel ensemble of processors, limiting the advantages of parallel architectures.

In search for higher speed and more cost effective designs, heterogeneity in parallel processing system is analysed in this paper as a viable approach to construct general purpose supercomputers. Heterogeneity has appeared at different levels in high performance system design [3]. In this work we consider heterogeneity at the system level, where classes of processors of different speed are dedicated to specific tasks and cooperate closely on the same job.

In this paper, we analyze heterogeneous systems with two different classes of processors. A heterogeneous supercomputer architecture consists of $P - 1$ identical processing elements and a single more powerful processing element. The former are called **parallel processors**, which execute the fraction of the computation that can be partitioned among the P processors. The latter processing element, called **sequential processor**, is designated to process the serial fraction of the computation. The rationale for this proposed architecture stems from the fact that more powerful processors can reduce the processing time of serial fractions and improve the speedup upper bound of parallel architectures. Heterogeneous parallel architectures make use of the best of two worlds; they combine the speed of powerful single processor with the unlimited growth of a set of homogeneous and cheap lower capacity processors.

3 Performance Models

The basic parameters and assumptions for the performance models used here are:

- P Number of processors.
- C Capacity (in MIPS) of each processor in the homogeneous case.
- C_s Capacity (in MIPS) of the sequential processor in the heterogeneous case.
- C_p Capacity (in MIPS) of each parallel processor in the heterogeneous case.
- λ Average job arrival rate (jobs/second).
- $\tilde{I}$ random variable that indicates the number of instructions of a job.
- F_s fraction of the total number of instructions of a job that must be executed serially.
- F_p fraction of the total number of instructions of a job that may be executed in parallel.

$\overline{I} = E[\tilde{I}]$ and $\overline{I^2} = E[\tilde{I}^2]$ are, respectively, the first and second moments of $\tilde{I}$.

Assumptions:

- Jobs arrive from a Poisson Process.
- The r.v. $\tilde{I}$ is exponentially distributed.

The performance measure of interest in all cases will be the average job response time, denoted by T.

3.1 Performance Model for the Centralized Computation Paradigm

Since the number of instructions $\tilde{I}$ is assumed to be exponentially distributed, the processing time of a job at the central facility is given by $\frac{\tilde{I}}{C}$ and is also exponentially distributed. Thus, T can be directly obtained from the results of the queue M/M/1 [11] as follows

$$T = \frac{\frac{\tilde{I}}{C}}{1 - \frac{\lambda \tilde{I}}{C}} \tag{1}$$

3.2 Performance Model for the Homogeneous Parallel Computation Paradigm

In the parallel computation case, the total processing time of a job is given by the sum of its sequential processing time (i.e. part of the job that cannot be broken up into parallel portions of code) and of its parallel part. The appropriate queueing model in this case is an M/G/1 model (see [11]) which requires the first and second moments of the total processing time.

Let $\tilde{t}$ be the random variable that measures the total processing time of a job, $\tilde{t}_s$ be the random variable that indicates the time spent to process the sequential part of the code, and $\tilde{t}_p$ the random variable that measures the time spent in the parallel part of the program. The time $\tilde{t}_p$ includes all the overhead due to synchronization, communication and contention for any shared resource. Hence,

$$\tilde{t} = \tilde{t}_s + \tilde{t}_p \tag{2}$$

Some relationships between the first and second moments of the above defined random variables are given below.

$$\overline{t} = \overline{t_s} + \overline{t_p} \tag{3}$$

$$\overline{t^2} = \overline{t_s^2} + \overline{t_p^2} + 2\,\overline{t_s}\,\overline{t_p} \tag{4}$$

Notice that equation (4) above assumes that $\tilde{t}_s$ and $\tilde{t}_p$ are independent random variables. This assumption takes into account that in a parallel system, probabilistic computing times for tasks are more appropriate when modeling the execution of real programs. The source of randomness in computing time are the hardware and software of the system, and the computation itself [8].

Let us first consider the case where the parallel system is made up of homogeneous processors. Hence,

$$\tilde{t}_s = F_s \frac{\tilde{I}}{C} \tag{5}$$

$$\tilde{t}_p = E(P) + S(P) \tag{6}$$

where $E(P)$ is a function that gives the total time to execute the parallel portion of a job and $S(P)$ is the time spent due to synchronization, communication and any other resource contention overheads. Three forms for $S(P)$ are discussed in [12]:

$$S(P) = c \log_2 P \tag{7}$$

$$S(P) = c\frac{P}{2} \tag{8}$$

$$S(P) = \frac{c}{2} P \log_2 P \tag{9}$$

We consider here the expression for $\tilde{t}_p$ given in (10), which assumes that the total number of instructions to be executed in parallel, $F_p \bar{I}$, are broken up into P instruction streams, each stream being executed by one of the P processors. The total time to execute the P parallel streams is the time it takes to process the longest one. The first and second moments of a random variable defined as the maximum of P exponentially independent and identically distributed random variables can be found in [16]. Hence,

$$\tilde{t}_p = \max\{\tilde{z}_1, \ldots, \tilde{z}_P\} \tag{10}$$

where the $\tilde{z}_i$'s are independent and identically distributed r.v.'s, with distribution equal to the r.v. $\frac{F_p \tilde{I}}{C P}$. Thus,

$$\overline{t_s} = F_s \frac{\bar{I}}{C} \tag{11}$$

$$\overline{t_p} = \frac{F_p \bar{I}}{C P} H_P = \frac{F_p \bar{I}}{C P} \sum_{i=1}^{P} \frac{1}{i} \tag{12}$$

Notice that the harmonic number H_P which appears in expression (12) is reasonably close to the natural logarithm of P [16].

From (3), (11) and (12) it follows that the average processing time is given by,

$$\bar{t} = F_s \frac{\bar{I}}{C} + \frac{F_p \bar{I}}{C P} \sum_{i=1}^{P} \frac{1}{i} \tag{13}$$

Now,

$$\overline{t_s^2} = \frac{F_s^2 \overline{I^2}}{C^2} \tag{14}$$

$$\overline{t_p^2} = \frac{F_p^2 (\bar{I})^2}{C^2 P^2} \left(\sum_{i=1}^{P} \frac{1}{i^2} + \left(\sum_{i=1}^{P} \frac{1}{i} \right)^2 \right) \tag{15}$$

Therefore the second moment of $\tilde{t}$ can be obtained directly from (4), (11), (12), (14) and (15) as indicated below.

$$\overline{t^2} = \frac{F_s^2 \overline{I^2}}{C^2} + \frac{F_p^2 (\bar{I})^2}{C^2 P^2} \left(\sum_{i=1}^{P} \frac{1}{i^2} + \left(\sum_{i=1}^{P} \frac{1}{i} \right)^2 \right) + 2 \frac{F_s F_p (\bar{I})^2}{C^2 P} \sum_{i=1}^{P} \frac{1}{i} \tag{16}$$

Finally, the average response time T can be given by

$$T = \bar{t} + \frac{\lambda \overline{t^2}}{2(1 - \lambda \bar{t})} \tag{17}$$

3.3 Performance Models for the Heterogeneous Parallel Computation Paradigm

The results of section 3.2 may be modified in a straightforward manner in order to account for a heterogeneous parallel architecture composed of a processor (generally more powerful than the remaining $P-1$ processors) which will be assigned to execute the sequential part of the computation.

Thus, the average processing time for the heterogeneous architecture is

$$\bar{t} = \frac{F_s \bar{I}}{C_s} + \frac{F_p \bar{I}}{C_p (P-1)} \sum_{i=1}^{P-1} \frac{1}{i} \tag{18}$$

and its second moment is given by

$$\overline{t^2} = \frac{F_s^2 \overline{I^2}}{C_s^2} + 2 \frac{F_s F_p (\bar{I})^2}{C_s C_p (P-1)} \sum_{i=1}^{P-1} \frac{1}{i} + \frac{F_p^2 (\bar{I})^2}{C_p^2 (P-1)^2} \left(\sum_{i=1}^{P-1} \frac{1}{i^2} + \left(\sum_{i=1}^{P-1} \frac{1}{i} \right)^2 \right) \tag{19}$$

Finally, the average response time T for the heterogeneous case is given by equation (17) where $\bar{t}$ and $\overline{t^2}$ are given by equations (18) and (19).

4 Cost Considerations

Issues of economies of scale in computing came to consideration after an intuitive statement made by Herbert Grosch and published in [4]. This statement, which became known as Grosch's Law, was later revisited [5] and re-revisited by Ein-Dor [1]. The essence of Grosch Law is that the average cost of a computer system is proportional to the capacity of the processor raised to the power 0.45. The proportionality constant depends on the family of the processor: supercomputers, large mainframes, small mainframes, minicomputers and microcomputers. So, in general

$$Cost(f, C) = K_f \, C^{0.45} \tag{20}$$

where $Cost(f, C)$ is the average cost of a system of family f and capacity C. K_f is a family dependent constant. Let D be the total system cost (in

US$1,000.00) and K_{Hm} the constant relative to the processor family used to implement a homogeneous parallel architecture. Thus

$$D = P\,K_{Hm}\,C^{0.45} \qquad (21)$$

From (21) and the results given in section 3 one can state the following theorem (see proof in the Appendix).

Theorem 1 *The response time is an increasing function of the number of processors for the parallel computation paradigm, provided that system cost is kept constant when the number of processors is increased (this implies that the capacity of each processor must be reduced).*

Theorem 1 implies that it is better performance-wise in parallel systems to have a smaller number of more powerful processors than a larger number of less powerful processors, provided that the system cost remains unchanged.

5 Modeling Results

Using the analytic models developed in section three we are now able to draw curves that compare the average response time of a transaction processing workload submitted to two different computer systems organized under the two basic computation paradigms. The total system cost is assumed to be the same for the two systems. The centralized system is assumed to be implemented by a single powerful processor from the family of supercomputers and the parallel architecture is built out of several microprocessors. The characterization of these families of machines was described in section four and is based on cost performance considerations.

Theorem 1 of section four establishes that in order to minimize response time in the parallel case, one should use less processors of higher capacity as opposed to more processors of lower capacity, provided that total system cost is kept constant. Assuming a fixed total system cost and based on the revised version of Grosch's Law [1], we can have a centralized system consisting of only one supercomputer of 25 MIPS or a parallel system composed of 1119 microprocessors of 0.8 MIPS each. Two classes of transactions are considered for the latter case. One exhibits the maximum possible degree of parallelism, i.e. the fraction of sequential processing (F_s) is equal to 0, and the other has an $F_s = 10\%$ of the total processing requirement. Figure 1 displays the variation of the average transaction response time as a function of the average transaction arrival rate.

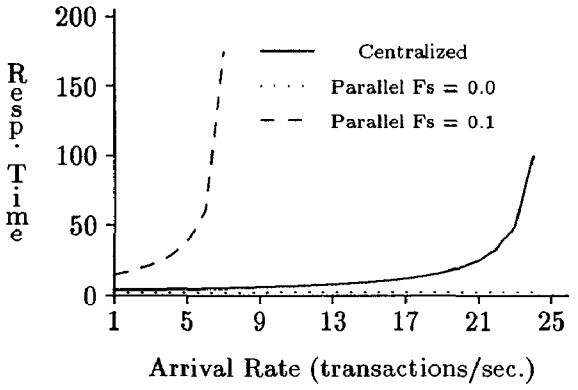

Figure 1: Performance of the Computation Paradigms

The best performance is obtained for the parallel computation model processing transactions with $F_s = 0$. However it is a well known fact that real programs from diverse application fields exhibit a non zero sequential processing fraction [7, 8]. The best case for real programs turns out to be the centralized model which exhibits a very low response time as compared to the other architectures. It should be noted that in order to upgrade the performance of a centralized system, one must use a more powerful processor, which implies in using more advanced component technologies. As indicated by Hack in [12], reductions in high performance machine cycle times have been coming at a much slower pace in recent years, a problem that is expected to become more accute as signal propagation constraints, which limit the physical size of the central processing unit, become the limiting technological factor. Optimistic projections suggest that cycle times in the vicinity of one nanosecond may be the limit for existing technologies. Current existing machines already present a cycle time close to that limit; NEC-SX3 [13] has a cycle time of 2.1 nanosecond. Therefore, the centralized approach for designing high performance machines is reaching its limit. Parallel processing has been proposed as a means to circumvent the technological growth limitations of single-processor architectures. Systems composed of several hundreds or even thousands of processors are becoming a commonplace these days [17].

Figure 1 shows also the behavior of a parallel sys-

tem when real programs, characterized by non-zero fraction of serial computing, are executed. As it can be noted in the figure there is a dramatic performance degradation in this case, which confirms the critical role of the sequential portion of the code on the overall performance of a system. This was first stated by Amdahl [9]. Therefore, the mere use of several identical processors in parallel does not appear to be the best solution to achieve high performance general-purpose computing. This paper proposes in the next section a heterogeneous approach to parallel processing.

5.1 Performance Comparisons

The performance metric used in the comparisons is the architecture speedup, which is defined as the ratio of the response time when executing a transaction on a homogeneous parallel architecture to the response time when the transaction executes on the heterogeneous parallel architecture of the same cost. The Processor Power Ratio (PPR) represents the degree of heterogeneity of the architecture. It is the ratio of the capacity of the sequential processor over the capacity of each identical parallel processor ($\frac{C_s}{C_p}$). This ratio equals to one in the homogeneous case.

Consider now a heterogeneous and a homogeneous parallel architecture of the same cost D, processing a workload defined by an average arrival rate λ and a computing demand $\overline{I}$, which is the average number of instructions to be executed in each transaction. This section analyzes the relative performance of the two computation paradigms.

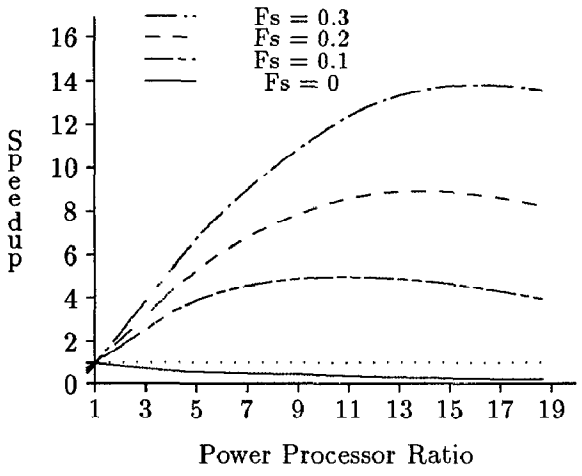

Figure 2: Heterogeneous versus Homogeneous

Figure 2 plots the architecture speedup as calculated by the queueing models (sections 3.2 and 3.3) as a function of the processor power ratio (PPR). In this figure, the average arrival rate is 3600 transactions per hour and the average number of instructions executed per transaction is one million. The homogeneous parallel architecture consists of 1106 identical 0.8 MIPS processors.

In order to specify the parameters of the heterogeneous architecture model (i.e., number of parallel processing elements and capacity and family of the sequential processor), a fraction of the total cost of a homogeneous system is assigned to the sequential processor of the selected family. Using Grosch's Law [1], we can determine the capacity of this processor, as well as the number of parallel processors to be removed. Varying the fraction of cost used to calculate the capacity of the sequential processor, several models of heterogeneous architecture are built and analyzed with the queueing models. Consider now a homogeneous parallel architecture and several configurations of heterogeneous parallel architectures, all of them with the same cost. Workloads with different fractions of serial processing are used to compare the types of architecture. First, notice that for workloads with non-zero serial processing fraction the heterogeneous architecture outperforms the homogeneous architecture, no matter the processor power ratio. For these workloads the speedup always remains greater than one and increases with the fraction of serial computing. The bigger the sequential fraction, the better the performance of the heterogeneous architecture.

The analysis now focus on the performance of the heterogeneous architectures when compared to that of centralized systems. For this purpose, consider a heterogeneous and a centralized architectures of the same cost D, processing identical workloads, specified by an average arrival rate λ and a computing demand $\overline{I}$. Although the workload places the same computing demand on both architectures, an additional parameter, the fraction of serial processing, is taken into consideration for the performance analysis of heterogeneous architectures.

The centralized architecture consists of a powerful 25 MIPS single processor, whereas the heterogeneous parallel architecture has three different models, with 440, 329, and 773 identical parallel processors respectively. The capacity of the sequential processor in each model is represented by the Processor Power Ratio (PPR), which are 9.86, 13.9 and 7.06 respectively.

Figure 3 plots the architecture speedup as calculated by the performance models, as a function of the serial processing of the workload. For the centralized

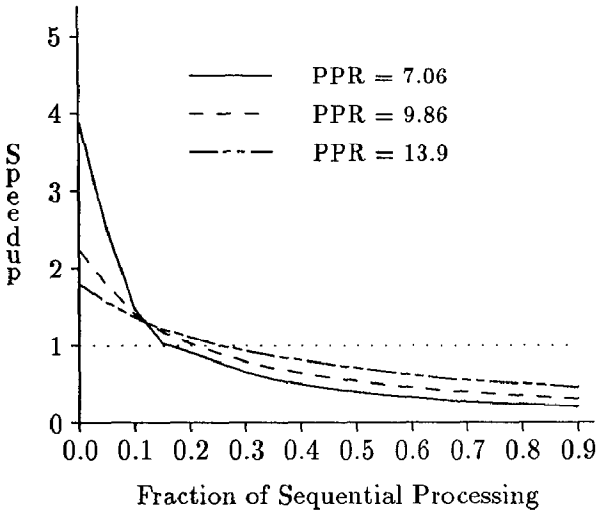

Figure 3: Heterogeneous x Centralized

architecture the response time does not vary with this parameter. The speedup plotted in this figure is the response time when a transaction is executed on the centralized system over the response time the when the transaction executes on a configuration of the heterogeneous parallel architecture.

The line corresponding to speedup one divides figure 3 into two regions: the one where the heterogeneous architecture performs better than the centralized system and the other region where the position inverts. The former region is located on the portion of the figure where the workload presents small fractions of serial processing. An interesting analysis consists of selecting a heterogeneous configuration and comparing its relative performance to a single processor of the same cost. Then, the performance of a cost equivalent homogeneous configuration is compared to the same single processor. Table 1 shows such a comparison for several values of the sequential processing fraction (F_s).

Table 1: Speedup of Parallel Architectures

Fraction(Fs)	*Speedup*	
	Homogeneous	Heterogeneous
0.00	4.9	2.3
0.10	0.28	1.4
0.20	0.12	1.1
0.30	0.07	0.8

For the unrealistic case of $F_s = 0$, the speedup of the homogeneous configuration is greater than that of the heterogeneous case. However, it can be seen from the table 1 that in all other cases, which resemble to real workloads, the heterogeneous configuration exhibits superior speedup.

6 Heterogeneity Issues

The models presented in this paper helped to show the cost-effectiveness of building heterogeneous supercomputers with two classes of processors. One can certainly imagine a more general situation in which there are several classes of processors of different speeds. An interesting and important problem that arises in such an environment is how to optimally schedule tasks to different processors. The criterium of optimality in this case would be the minimization of the execution time of a parallel application running on the architecture.

A parallel application is composed of a certain number of tasks which must be executed according to a ceratin precedence order, specified by a task graph [15]. Since task execution times are not deterministic, due to several delays imposed by contention on shared resources, such as common memory, or intercommunication network, it is not feasible to imagine scheduling algorithms that will always give the optimum schedule. Therefore, one must settle for near-optimal solutions.

The scheduling problem has been discussed in homogeneous environments in the work of Sevcik [15]. In a heterogeneous architecture, the problem becomes more complex due to the even greater diversity of scheduling alternatives, created by several classes of processors with different speeds. Problems such as scheduling of concurrent tasks, parallel algorithm design for heterogeneous architectures, to name a few, are entirely open research topics.

7 Conclusions

Amdahl's Law [9] has demonstrated that the serial fraction of processing dominates the execution time for any large parallel ensemble of processors, limiting the advantages of parallel supercomputers.

In search for higher speed and more cost effective designs, heterogeneity [3] in supercomputer architectures was introduced in this paper as a viable approach to construct general purpose high performance parallel systems. A heterogeneous parallel architecture consists of $P-1$ identical processing elements and a single more powerful processing element. The rationale for this proposed architecture stems from the fact that more powerful processors

can reduce the processing time of serial fractions and improve the upper bound speedup of parallel architectures. Heterogeneous parallel architectures make use of the best of two worlds; they combine the speed of a powerful single processor with the unlimited growth of a set of homogeneous and cheap lower capacity processors.

All performance comparisons carried out in this paper consider that the cost of the compared systems is the same. Grosch's Law, in its revised version [1], was used in order to relate cost and capacity of processors of the same family. Queueing models were developed in order to analyze the performance of homogeneous and heterogeneous computation paradigms. The results show that the heterogeneous architecture exhibits higher speedup when compared with cost equivalent homogeneous architectures. Results were derived and presented for various classes of applications characterized by different degrees of parallelism. Also, different configurations of heterogeneous architectures, represented by sequential processors of different capacities, were analyzed.

The results of this paper show that heterogeneity in supercomputer architectures is a quite promising design approach that deserves further investigation.

References

[1] Ein-Dor, Phillip, *Grosch's law re-revisited: CPU power and the cost of computation*, Communications of the ACM, Vol. 28, no. 2 (Feb 1985), pp 142–151.

[2] Menasce, Daniel and Almeida, Virgilio, *Analytic Models of Supercomputer Performance in Multiprogramming Environments*, The International Journal of Supercomputer Applications, Vol. 3, No. 2, Summer 1989, MIT Press.

[3] Ercegovac, Milos D., *Heterogeneity in supercomputer architectures*, Parallel Computing 7 (1988), North-Holland, pp 367–372.

[4] Grosch, H. A., *High speed arithmetic: the digital computer as a research tool*, J. Opt. Soc. Am. Vol. 43, No. 4, April 1953.

[5] Grosch, H. A., *Grosch's law revisited*, Computerworld, Vol. 8, April 16, 1975.

[6] Heidelberger P., *A performance comparison of multimicro and mainframe database architectures*, IEEE Transactions on Software Engineering, Vol. 14, No. 4, April 1988.

[7] Eager, D., Zahorjan J., Lazowska, E. *Speedup versus efficiency in parallel systems*, IEEE Transactions on Computer Systems, Vol. 38, No. 3, March 1989.

[8] Flatt, H., Kennedy, K. *Performance of parallel processors*, Parallel Computing, No. 12, 1989, North Holland.

[9] Amdahl, G. *Validity of the single processor approach to achieving large scale computing capability*, in: Proc. AFIPS Spring Joint Comp. Conf. 30, 1967.

[10] Kleinrock, L. *Distributed Systems*, Computer, Vol. 18, No. 11, November 1985.

[11] Kleinrock, L. *Queueing Systems, Volume I: Theory*, Wiley-Interscience, 1975.

[12] Hack, J. *On the promise of general purpose parallel computing*, Parallel Computing, 10 (1989), North Holland, pp 261-275.

[13] *Vector Register*, Vol. 2, No. 5, June 1989, Institute for Supercomputing Research, Japan.

[14] Hockney, R. and Jesshope, C. *Parallel Computers 2*, Adam Hilger, Bristol and Philadelphia, 1988.

[15] Sevcik, K. *Characterizations of parallelism and their use in scheduling*, Performance Evaluation Review, Vol. 17, No. 1, May 1989.

[16] Trivedi, K. *Probability and Statistics with Reliability, Queueing, and Computer Systems Applications*, Prentice Hall, 1982.

[17] Bell, Gordon *The future of high performance computers in science in engineering*, Communications of ACM, Vol. 32, No. 9, September 1989.

[18] Wallach S., *What will the next generation supercomputer look like?*, Supercomputing Review, January 1990.

Appendix: Proof of Results

Proof of Theorem 1

From equation (17) we know that the average response time T is given by

$$T = \bar{t} + \frac{\lambda \overline{t^2}}{2(1 - \lambda \bar{t})} \qquad (22)$$

Let φ denote a constant value. Arithmetic operations involving φ and other constants will give as result another constant which will be denoted by φ no matter what its value is.

From the constant cost assumption we have that

$$P_p K_p C_p^{0.45} = \varphi \tag{23}$$

Thus, from (23) we have that

$$C_p = \frac{\varphi}{P_p^{2.22\ldots}} \tag{24}$$

In order to prove the theorem let us prove the following lemmas:

Lemma 1 *$\bar{t}$ is an increasing function of P_p.*

From equation(13), we have that

$$\bar{t} = \frac{\varphi}{C_p} + \frac{\varphi}{C_p P_p} H_{P_p} \tag{25}$$

From (24) and (25) it follows that

$$\bar{t} = \varphi P_p^{2.22\ldots} + \varphi P_p^{2.22\ldots} H_{P_p} \tag{26}$$

The lemma follows directly from equation (26)

Lemma 2 *$\overline{t^2}$ is an increasing function of P_p.*

From equation (16) we have that

$$\overline{t^2} = \frac{\varphi}{C_p^2} + \frac{\varphi}{C_p^2 P_p^2}\left(H_{P_p}^2 + \sum_{i=1}^{P_p} \frac{1}{i^2}\right) + \frac{\varphi}{C_p^2 P_p} H_{P_p} \tag{27}$$

From (24) and (27) we have that

$$\overline{t^2} = \varphi P_p^{4.44\ldots} + \varphi P_p^{2.44\ldots}\left(H_{P_p}^2 + \sum_{i=1}^{P_p} \frac{1}{i^2}\right) + \varphi P_p^{3.44\ldots} H_{P_p} \tag{28}$$

The lemma follows directly from equation (28).

The theorem proof follows immediately from the inspection of equation (22) and lemmas 1 and 2. Q.E.D.