12-13-2002

An Analysis for Evaluating the Cost/Profit Effectiveness of Parallel Systems

Maria Teran

AN ANALYSIS FOR EVALUATING THE COST/PROFIT EFFECTIVENESS OF PARALLEL SYSTEMS

By

Maria Teran

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science

Mississippi State, Mississippi

December 2002

AN ANALYSIS FOR EVALUATING THE COST/PROFIT EFFECTIVENESS

OF PARALLEL SYSTEMS

By

Maria Teran

Approved:

Edward A. Luke Assistant Professor of Computer Science (Major Professor)	Julian E. Boggess Associate Professor of Computer Science Graduate Coordinator Department of Computer Science
Ioana Banicescu Associate Professor of Computer Science (Committee Member)	Donna Reese Associate Professor of Computer Science (Committee Member)

A. Wayne Bennett
Dean of the College of Engineering

Name: Maria Teran

Date of Degree: December 13, 2002

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: Dr. Edward A. Luke

Title of Study: AN ANALYSIS FOR EVALUATING THE COST/PROFIT EFFECTIVENESS OF PARALLEL SYSTEMS

Pages in Study: 125

Candidate for Degree of Master of Science

A new domain of commercial applications demands the development of inexpensive parallel computing platforms to lower the cost of operations and increase the business profit. The calculation of returns on an IT investment is now important to justify the decision of upgrading or replacing parallel systems.

This thesis presents a framework of the performance and economic factors that are considered when evaluating a parallel system. We introduce a metric called the cost/profit effective metric, which measures the effectiveness of a parallel system in terms of performance, cost and profit. This metric describes the profit obtained from the performance of three different domains for scaling: speed-up, throughput and/or scale-up. Cost is measured by the actual costs of a parallel system. We present two cases of study to demonstrate the application of this metric and analyze the results to support the evaluation of the parallel system on each case.

DEDICATION

To my family, close friends and most of all to Gianni, this work is a result of the motivation and support he put in me. Again, thanks a lot.

ACKNOWLEDGMENTS

First of all, I would like to thank my major professor Dr. Edward Luke for the help, guidance and support provided in the realization of this work. Dr. Luke motivated me to continue my research and helped me to give shape to a very broad area of study. His ideas and advices allowed me to continue in times when I felt discouraged. I thank him for his important contribution in this thesis.

I will also like to thank the systems administrators at the Engineering Research Center(ERC) in Mississippi State University and the Cornell Theory Center (CTC) at Cornell University, for providing me with useful inside information about their systems platforms. Specially I extend my acknowledgments to Roger Smith from the ERC and to Paul Redfern from the CTC.

Finally I would like to thank my family and close friends for their support and concern at any time. Specially to Giovanni Modica, my right and left hand through this master program.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

LIST OF SYMBOLS, ABBREVIATIONS, AND NOMENCLATURE

The following is a list of symbols used in this document.

$C_A(S,t)$ Architecture cost function.

$C_d(S,t)$ System downtime cost function.

$C_H(S,t)$ Hardware cost function.

$Cost(S_p,t)$ Cost function.

$C_S(S,t)$ Software cost function.

$C_Su(S,t)$ Staff and Support cost function.

$C_U(S,t)$ Utility cost function.

E Efficiency.

f_d Factor of depreciation.

K Positive constant.

M_{S_i} Marginal productivity of performance.

p Number of processors.

Pro Production function.

$Profit(S_p,t)$ Profit function.

$P(S_p)$ Performance from system configuration.

s Speedup.

S_p System configuration with p processors.

T_o Overhead function.

T_p Execution time of a parallel system.

T_s Execution time of a serial system.

W Problem size.

Γ_{factor} cost/profit factor.

Γ_{profit} cost/profit effective metric.

Γ_{opt} Optimal effectiveness metric.

$\varphi(P(S))$ Unit gain rate obtained from the performance of the system configuration.

$\Sigma_c(S,t)$ Explicit cost function.

$\Theta_c(S,t)$ Opportunity cost function.

$\rho_u(S,t)$ Under utilization cost function.

CHAPTER I

INTRODUCTION

In its early beginnings, high performance computing (HPC) was oriented toward government and scientific applications without the concern of financial or economic implications inherently present in parallel systems. Nowadays, the parallel computing market has changed and HPC can now be deployed on commodity processors using advanced networking technology to create inexpensive parallel computing platforms to lower the cost of operations and increase profit.

Customers and vendors are now concerned with the relation between cost, performance and profit for evaluating parallel systems. There is no single approach to compare parallel systems. The use of benchmarks (*i.e.*, LINPACK benchmark, NAS Parallel benchmarks, ScaLAPACK, etc.) is a very common choice for customers and vendors when evaluating several systems. Moreover, organizations like the *Top 500* list[1] are also used to consider the performance evaluation of a parallel system based on peak theoretical speeds. However, comparisons surrounding the capabilities and power of parallel systems do not involve other factors that are present in the real scenario that many Information Technology (IT) managers have to confront, making it difficult for them to answer questions like "Given

[1] `http://www.top500.org`

a particular computational task, a limited budget and an extensive technological market, what parallel system will best suit the needs for achieving demanding business goals?". Or, "when is it convenient to replace or upgrade the actual system to increase the business profit and/or lower operational costs?"

From a financial and economic point of view, the information technology department of many companies reflects costs rather than benefits. Investments in IT infrastructure are done mainly on demand, and there is an interest in developing more accurate, predictable, and representative analyses to evaluate these investments from a business perspective. These aspects, and the broad market of technology, led to the application of economic analyses in the computational industry. Some of the standard models of economics[11, 20] have been proposed to measure the cost assessments and return on investment in computational systems, but no single methodology exists that can help to determine and measure the business benefits obtained from investing in parallel systems.

From scientific point of view, many researchers have proposed metrics for analyzing the performance and scalability of parallel systems [34, 37], but these metrics cannot make a sufficient justification for deciding on the investment of such systems, although they acknowledge through measurements the role of cost in high performance computing [52, 86].

Ideally, the integration of cost and profit models should lead to a more general way to evaluate parallel systems. Toward this goal some authors have presented different initiatives. For example, the cost-effective metric [85] proposes a direct relation of cost to

performance by implying that a system is cost-effective whenever it has a higher performance/cost ratio, and by this relation, it concludes that a system is cost-effective if *speedup* exceeds *costup*. The drawback of this metric is that is too pessimistic in judging the financial justification of parallel systems.

Another metric related to this work was proposed in the profit-effective analysis of parallel computation [86]. This study proposes a metric (*profitup*) which measures the relation of a parallel system and a serial system in terms of profit and performance. This work presents an extension of the cost-effective metric, and relates the profitup metric to the costup metric by stating that a parallel system is profit effective if the profitup metric exceeds the costup metric.

This thesis provides a framework to analyze the factors that are involved when evaluating a parallel system in terms of performance, cost and profit. The framework includes the definition of parallel system effectiveness from the business perspective based on system productivity. It also identifies the factors that are considered when evaluating a parallel system, which we categorized in four groups: potential, financial, environmental, and cultural. The discussion of two approaches for scalability in parallel systems (scale-in and scale-out), and the analysis of the depreciation factor in parallel computing, are also included in this framework.

We also introduce as part of this framework, the cost/profit effective metric. We use this metric to quantify the effectiveness of a parallel system in terms of performance, cost and profit. We apply this metric to two case studies, the Cornell Theory Center (CTC) and

the Engineering Research Center (ERC). We conclude by showing how these results can be used to evaluate the decision regarding upgrading or replacing a parallel system.

1.1 Motivation

As previously mentioned, the HPC market is changing and organization's investment toward this market is recently being analyzed. Technology advances have produced commodity processors that, combined with high performance networks, are now competing with traditional parallel systems.

IT managers have to manage the cost of operations with very high expectations to meet their goals and handle aggressive markets to obtain the best technology that will apply to their needs for today and tomorrow. For example, in Figure 1.1 we can observe a typical scenario for an IT manager in an organization. On one side he or she has to manage all the resources to achieve the organization goals (heterogeneous Datacenter, demanding users, competitive market, limited budget, profit goals, etc.) and on the other side he or she needs to choose from a broad market of technologies the best solution to meet the demands of the organization. This task gets more complicated when there is no single method or rule of thumb to follow when it comes to finding a criteria for deciding on a particular system.

Furthermore, most known metrics for evaluating parallel systems concentrate on the raw performance of computation, rather than on the economic impact to the business. As Yan and Zhang stated "Because high performance has strongly motivated parallel computing research and development for advanced applications, profit has not been a real

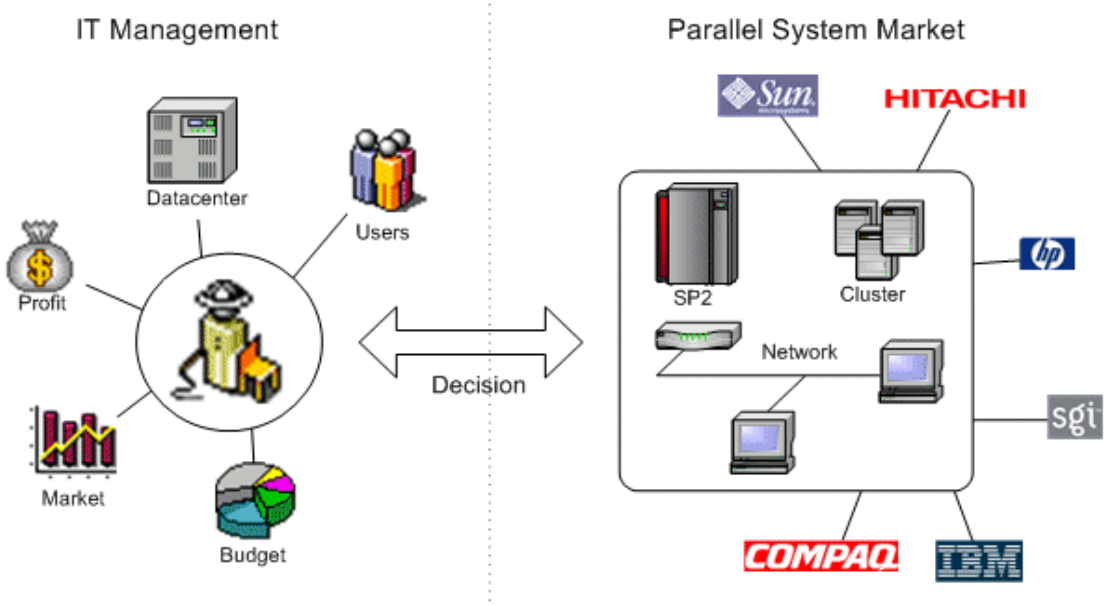

Figure 1.1 IT managers task

concern," [86] but now HPC has expanded to many markets where profit gain is a main concern.

For these reasons, we investigate a framework that analyzes the evaluation of parallel systems from a business perspective. This framework comprises the main factors, in terms of performance, cost and profit, that must be considered deciding between upgrading or replacing a parallel system. And by the use of the cost/profit effective metric we are able to quantify the impact that the productivity of a parallel system brings to an organization.

1.2 Thesis

As a consequence of the expansion of high performance computing to other areas besides research and development of advanced applications, there exists a broad market of

HPC platforms that range from state-of-the-art architectures to inexpensive parallel systems composed of commodity processors with advanced networking technology.

Customers and vendors frequently compare parallel systems according to the relation between price and performance. The idea of obtaining performance at any cost has changed. Many IT departments now need to justify the investment to HPC systems. Some economic models have been introduced to analyze the cost of ownership of computing systems [11, 20], and there has been some interesting work [85, 86] which describe the effectiveness of parallel systems in terms of cost and profit.

This type of analysis can support IT managers to justify the investment on a particular architecture and to decide between upgrading or replacing its infrastructure. We believe that the creation of a framework that unifies all these economic and performance analyses, will give the parallel community a means to evaluate parallel systems in terms of performance, cost and profit. Our main objectives are as follows:

- Identify the main factors to be considered when evaluating parallel systems in terms of performance, cost and profit.
- Introduce a metric for measuring the factors that are presented by the framework and describe how this metric can be applied to support the decision of upgrading or replacing a parallel system infrastructure.
- Apply this metric to a case study where the decision of upgrading or replacing a parallel system is required and show the analysis of the results.

Our goal is to combine all the objectives mentioned above into a framework to evaluate parallel systems in terms of performance, cost and profit. We believe that the analysis of this framework can be used by decision makers in the IT environment to evaluate and

compare among the different vendors in the parallel market and can also be used by the vendors as a competitive tool to offer better solutions to their customers.

1.3 Organization

The rest of the document is organized as follows:

- Chapter II presents background information and discusses previous work in the area of performance and cost analysis for parallel systems. Important terminology and concepts, such as speedup, efficiency, cost and performance, are presented. Standard models of economics, such as Total Cost of Ownership (TCO), Return on Investment (ROI) and Total Economic Impact (TEI), are also formally defined.
- Chapter III talks about the development of a framework for evaluating parallel systems in terms of performance,cost, and profit. In this chapter we also introduce the definition of the cost/profit effective metric.
- Chapter IV presents how to determine the production function of a parallel system based on its performance. Three scaling paths are presented for the calculation of system performance: speed-up, throughput, and scale-out.
- Chapter V defines the cost function based on the concept of Total Cost of Ownership (TCO). It considers the different cost factors obtained from different parallel architectures
- Chapter VI presents the case study of two academic organizations, the Cornell Theory Center (CTC) and the Enigineerin Reserch Center (ERC). The cost/profit effective metric is applied and the results are analyzed to support the decision of upgrading or replacing the parallel system platforms on in each case.
- Chapter VII provides conclusions about the concepts presented in this document.

CHAPTER II

DEFINITIONS AND BACKGROUND

In this chapter we present a literature review and provide background information on different topics covered by this thesis. First, we will present the evolution of parallel computing and the importance it has gained over the years. We will follow with a discussion of the evaluation of parallel systems. Next, we will present some of the well-known metrics used to evaluate the performance of parallel systems and conclude the chapter with some economic background on different models that have been used to analyze the financial impact of computing systems.

2.1 Evolution and Importance of Parallel Systems

The primary reason for developing parallel computing was the necessity to solve larger and more complex problems. The limitations found in traditional serial computers (*i.e.*, CPU memory, I/O bandwidth, etc.) were an obstacle to the deployment of sophisticated applications used to solve problems within a limited amount of time [13]. For example, weather forecasting is a time-critical application that requires HPC. Also, the emergence of applications that simulate complex problems, that were too large and too expensive to test in real life, required the use of high-speed computation [83]. For example, crash

simulation in the car industry. HPC has now moved to other areas besides government and academic research. Many health care, telecommunications, manufacturing, and oil and gas applications rely on HPC systems.

There are many ways to construct parallel systems, their architecture varies according to the following dimensions: mechanism, address-space organization, network and processors granularity [34]. Flynn's taxonomy (based on mechanism) and memory organization (based on address-space organization), are perhaps the two most referenced classifications of parallel systems [29].

2.1.1 Flynn's Taxonomy

There are four classification to Flynn's taxonomy:

SISD (Single Instruction Single Data): Corresponds to traditional systems. It is constituted by single processor and memory, which take a single stream of instructions and executes them serially on a single stream of data. For example, some mainframes are formed by a group of SISD machines that operate independently on different data space. Other examples of this type of architecture are workstations from HP, SUN Microsystems and Compaq [83].

SIMD (Single Instruction Multiple Data): Machines built under SIMD architecture use several processing units to execute the same instruction under different data. The manipulation of vectors and matrices are a common example of the usage of these systems.

For example, systems that have parallel vector processing (PVP) like the vector supercomputers.

MISD (Multiple Instruction Single Data): There are no true implementations of MISD. Multiple instructions work on a single data.

MIMD (Multiple Instruction Multiple Data): At any time, many processors may execute many instructions in different types of data. This type of architecture reflects the real concept of parallelism, since many subtasks can be executed concurrently in order to improve the time to solve the main task.

2.1.2 Memory Organization

Memory Organization is divided in two main classifications [34]:

Shared memory systems: This type of architecture is implemented with a common memory that can be accessed and shared by all the processors. It also provides transparency since there is no concern about where data is stored. There are two subclasses of shared memory systems: Uniform Memory Access (UMA) and Cache Coherent-Non Uniform Memory Access(CC-NUMA). In the UMA class the time it takes a processor to access any word in the shared memory is identical. Examples of these architectures are Compaq's GS60/140 series and the Sun E10000/15000 SunFire and StarFire series. These systems are also commonly known as Symmetric Multiprocessor Systems (SMP). In CC-NUMA, each processor has a local memory besides the access to remote memory to

improve processor-memory bandwidth. An example of this architecture is the HP V2500 series.

Distributed memory systems: In this type of architecture each processor unit has its own local memory and is connected to a fast interconnection network (bus, crossbar, switch, etc). The user must have the knowledge of where the data is in order to make the request of retrieval. These systems are often called multicomputers or Massive Parallel Processing (MPP). They are different from CC-NUMA architecture because they communicate through message passing (explicit approach) while the latter has the hardware support to communicate to other processors memory without message-passing (implicit approach). Examples of distributed systems IBM RS/6000 SP series and commodity processor cluster systems.

Figure 2.1 depicts the different classes of architectures according to Flynn's taxonomy and the memory address space organization. As mentioned previously, there exist many other ways in which parallel computers are classified, but we make reference these two because they are the most common and general among the categorizations of parallel architectures.

A parallel system is defined as the combination of algorithms and the architectures where they are implemented [34]. The lack of parallel standards generates a dependency between parallel algorithms and some architectures. We can observe from the broad classification of parallel systems that different algorithms can exist for different architectures, and some will behave and perform better according to the architecture they are imple-

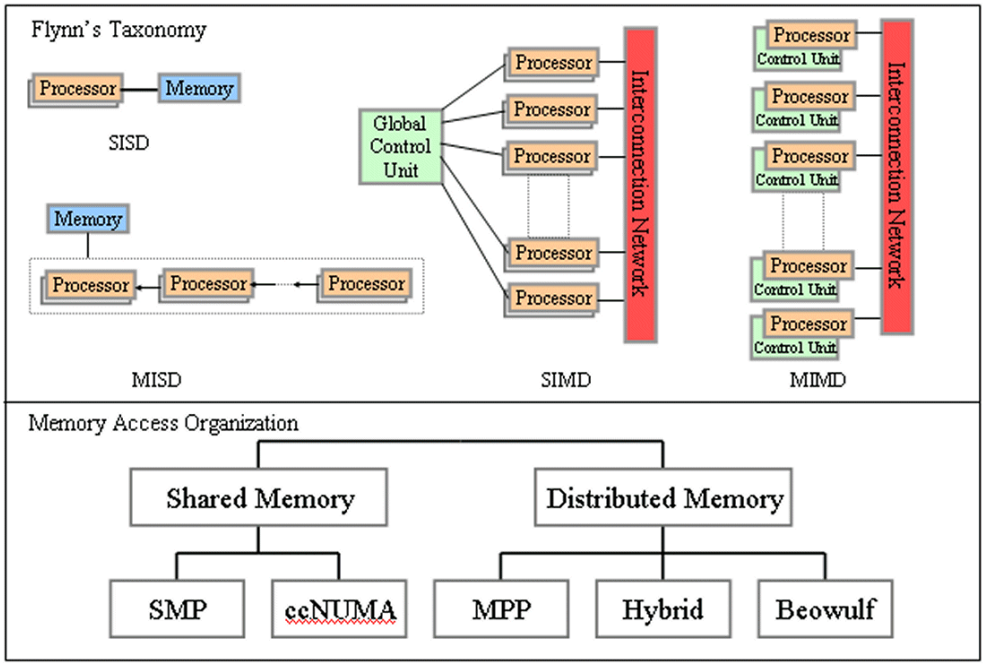

Figure 2.1 Classification of parallel architectures

mented on. For example, in previous years problems such as complex mathematical equations and vector processing were implemented in specialized vector processing architectures, as we can observe in Figure 2.2. The systems dominating the market in 1994 were the vector processor systems, with almost 50% of the market share. The areas that exploited these types of problems were oriented towards research and academic computing, as we can see in Figure 2.3.

Although the information provided in these graphs is not up to date, we are more interested in observing the relations and trends that exist between applications, architectures and the areas where they are exploited. Figure 2.2 and Figure 2.3 are results from a study

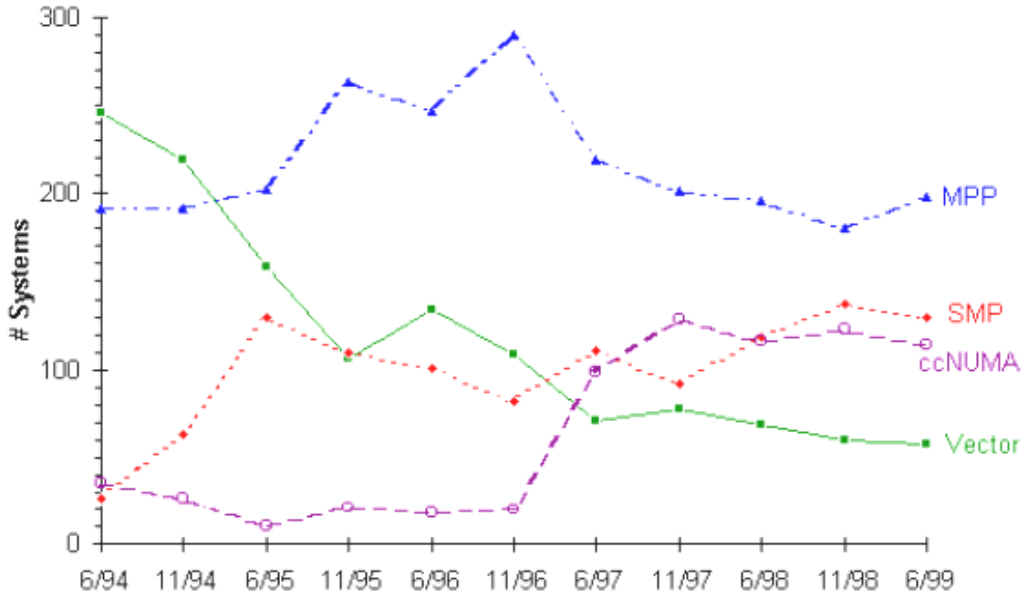

Figure 2.2 Top 500 List reports from 1994 to 1999 according to the architecture.

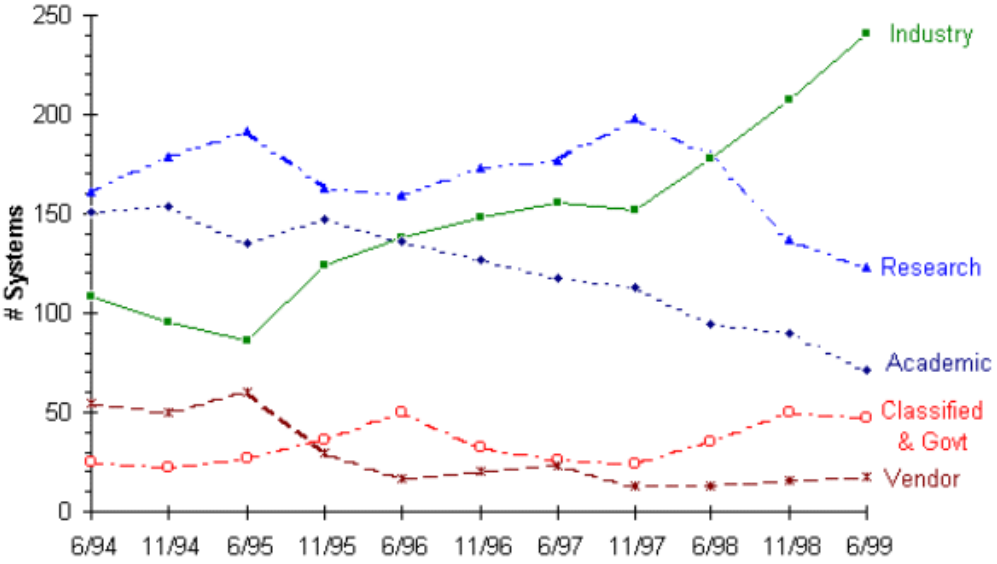

Figure 2.3 Top 500 List reports from 1994 to 1999 according to the areas of application.

made by the firm D.H.Brown Associates [1] according to the information provided by the Top 500 list organization [2]. Recent reports are not publicly available.

Since 1993 the Top 500 list organization assembles and maintains a list of the 500 most powerful computers on site in the world. The list is generated twice a year and the computers are ranked by their performance on the LINPACK Benchmark [22], described in section 2.2.2 of this chapter.

As technology advanced and other areas started to develop, the scene for manufacturers of Parallel Vector Processor(PVP) systems, began to change. The demand for MPP systems overgrew the demand for PVP systems. MPP systems dominated the market in 1995 with 40% of install base, while the PVP systems install base dropped to 30% of the share. On the other hand, we can observe from Figure 2.3 that in 1995 the industry area began to develop and utilize high performance applications to increase productivity, which created a growth of the installed base of parallel systems in this area.

At present, industry is dominating the number of installed sites of parallel systems, as shown in Figure 2.3. The MPP architectures are also dominating the market as the architecture of choice for the deployment of parallel applications (Figure 2.2). With these developments in consideration we can see that there exists many trends concerning the functionalities and capabilities of parallel systems, and that customers can be distracted by these facts when determining a proper evaluation of parallel systems [75].

---

[1] `http://www.dhbrown.com`

[2] `http://www.top500.org`

2.2 Evaluation of Parallel Systems

As a consequence of the growth of parallel computing, there is a need for evaluating parallel systems to answer questions such as: "How will the system perform?" and "On what basis should performance be measured?" These questions relate to a performance evaluation study.

As we have seen in the previous section, there are many architectural approaches. Some applications will perform well for a particular architecture while others will perform poorly on the same architecture. This contradictory behavior can be the product of the poor design of some parallel applications that do not portably exploit the resources provided by the architecture. In general, parallel applications are not easy to port. As Valiant noted[82], "the success of the Von Newmann model of sequential computation is attributable to the fact that it is an efficient bridge between software and hardware." This bridge has not yet been found in the design of parallel systems [72]. Although many models have been proposed [18, 30, 82], there is still a struggle to find a universal model or at least a small number of fundamental models for parallel computation.

According to the Center for Parallel Computing Research (CPCR)[3] [10], the high-performance computing community, need to develop a standardized, rigorous, and scientifically methodology for studying the performance of high-performance computer systems. This methodology would help to:

[3] `http://www.crpc.rice.edu/CRPC.html`

- Increase the understanding of HPC systems, at a low-level hardware and software perspective. Also at a high-level, total system performance perspective
- Assist customers of HPC systems to select the system that best suites their needs
- Reduce the amount of time and resources vendors spent in implementing multiple, redundant benchmarks
- Provide valuable feedback to vendors for future products
- Improve supercomputer performance analysis as a serious scientific discipline
- Reduce confusion in the high-performance computing literature

2.2.1 Performance Evaluation

Performance evaluation can be defined as the quantifying values assigned to the behavior of a system under certain workload and with a limited amount of resources. There is no single definition for performance according to several authors [21, 33], but some factors must be taken into consideration [43], such as functionality, reliability, speed and economics.

Functionality relates to the successful achievement of the basic demands for which the system was built. The reliability of the system depends on the probability of an error in its outcomes. If the probability is too high the system should be adjusted to satisfy reasonable levels. The functionality and reliability factors are commonly studied in the design stage of a parallel system, and then the performance evaluation is demonstrated by analytical modeling or simulation techniques.

The speed should represent how fast and efficient the system can complete its work. Speed, is the most common factor for performance evaluation, and it has multiple repre-

sentations. The cost of the system is reflected by the economic factors, *i.e.* the design and implementation of a system should be driven by the lowest cost possible that does not sacrifice reliability, functionality and speed.

We are most interested in the speed and the economic factors, which are commonly viewed after the system is built and implemented. These factors are relevant when evaluating parallel systems. To compare and evaluate the performance of several parallel systems it is necessary to choose certain metrics. The creation, selection and usage of metrics are highly problem-oriented. Some studies have been done toward selecting proper metrics [47]. The services that are offered by a system are a common measure of its performance. According to Hu and Gorton [43], any system offers three types of outcomes: *a*) the correct response to a service request, *b*) the incorrect response to a service request, and *c*) no response to a service request. The first case corresponds to metrics that are widely used in performance studies such as throughput, utilization, response time, and turnaround time. The second and third cases relate to the measure of the probability of errors and failures (*e.g.* measure of mean time between failures of a system). In general, metrics should be simple (not too many parameters involved), unique (choose the one that is most representative of those that have the same effect), and representative (should be able to describe the performance of interest).

2.2.2 Benchmarks

According to the Standard Performance Evaluation Corporation (SPEC) [4] a "benchmark" is a test or set of tests designed to compare the performance of one computer system against the performance of others. A realistic test should use programs designed to simulate a real workload. Benchmarks are a common tool for evaluating and comparing systems. For parallel systems there exists a variety of benchmarks that test their performance. The most widely known are:

- NAS Parallel Benchmarks (NPB) [5]: The NPB's are constituted by eight programs, derived from Computational Fluids Dynamic (CFD) applications. This programs consist of five kernels and three pseudo-applications. NPB one, are the original pencil and paper benchmarks, while NPB two are MPI based implementations. Serial implementations are also available.
- Linpack Benchmark [23]: The Linpack benchmark is a collection of Fortran subroutines that analyze and solve various systems of simultaneous linear algebraic equations. The subroutines are designed to be completely machine independent, fully portable, and to run at near optimum efficiency in most operating environments.
- The Parkbench Parallel Benchmarks: The Parkbench parallel benchmarks are a collection of 21 programs some of which are taken from the NPB and Genesis benchmarks. They are implemented in MPI and PVM.
- The ScaLapack Benchmark [7]: The ScaLapack benchmarks is constituted by library of high performance linear algebra routines for distributed message passing MIMD computers and networks of workstations supporting PVM [31] and/or MPI [59]. It is a continuation of the LAPACK [4] project, which designed and produced analogous software for workstations, vector supercomputers, and shared-memory parallel computers.

[4] `http://www.specbench.org/`

2.2.3 Scalability Analysis

The evaluation of scalability can also be used to determine the performance of parallel systems, although there is no commonly accepted definition yet in parallel computing [41]. In theory, the scalability of a parallel system measures the capacity to effectively utilize an increasing number of processors. There is an interest in the study of scalability analysis for parallel systems. From the work of [49] we find a summarization of where scalability analysis is found to be most useful:

- For problems where there exist different constraints on the growth of the workload and the number of processors.
- Performance prediction from known performance on fewer processors.
- Determining the optimal number of processors, for fixed problem size.
- Predicting the impact of changing hardware technology.

2.3 Metrics for Evaluating the Performance of Parallel Systems

In the previous section we discussed the importance of selecting proper criteria for evaluating parallel systems. This criteria is what we commonly call metrics. The selected metrics should take a collection of parameters that affect the system performance and quantify them in a way that can reflect the system's actual and future behavior. Many metrics have been proposed to measure the performance of parallel systems. We describe the most widely known and used for the effect of this study.

In order to formally define these metrics we introduce some notation:

- T_s: serial execution time of a serial algorithm.

- T_p: parallel execution time of a parallel algorithm.
- p: number of processors.

2.3.1 Speedup

Speedup represents how much performance gain is achieved by parallelizing a given application over a sequential implementation [34]. This performance gain is characterized by speed, so in essence speedup measures how much speed is gained by parallelizing an application over its sequential version. speedup is expressed by the ratio of the time to solve a problem on a serial system (represented by the serial execution time) to the time to solve the same problem on a parallel system (represented by the parallel execution time) [38, 49].

$$s = \frac{T_s}{T_p} \tag{2.1}$$

T_s represents the best serial execution time given in a serial computer, meaning that another serial algorithm with a better serial execution time does not exist. In theory, speedup can never exceed the number of processors. This limitation is imposed by the sequential fraction inherently present when parallelizing any application; this fraction provides an upper bound for speedup as described by Amdahl's Law [3]:

$$s = \frac{1}{f_s + \frac{(1-f_s)}{p}} \tag{2.2}$$

In Equation 2.2, f_s corresponds to the serial fraction of the application and $(1 - f_s)$ corresponds to the fraction that can be parallelize among p processors. As the number of processors p increases the fraction that can be parallelize decreases ($\frac{(1-f_s)}{p} \approx 0$), as a result the speedup s is bounded by the serial fraction $O(\frac{1}{f_s})$. When speedup is equal to the number of processors, it is called ideal speedup.

2.3.2 Efficiency

Efficiency is a measure of the time that a processor has done useful work; it is defined as the ratio of speedup to the number of processors. Efficiency is represent as [34, 38, 49],

$$E = \frac{s}{p} = \frac{T_s}{pT_p}, \tag{2.3}$$

where $0 < E \leq 1$. The analysis of efficiency allows us to study the percentage of the time the processors are performing other tasks not related to computation, such as communication, synchronization, etc. These other tasks correspond to the overhead of the total time of the parallel execution. When the parallel system achieves an efficiency of 1, then it has an ideal speedup. If efficiency can be maintained while adding more processors, we have linear speedup and a scalable system. This result, in practice, is difficult to achieve since on average, with the increase of speedup comes a decrease in efficiency. Hence, some kind of tradeoff is necessary to obtain the best results [24].

2.3.3 Isoefficency Metric

Isoefficiency was first introduced by Kumar,Gupta and Grama in 1993[35]. The isoefficiency metric is used for scalability analysis of parallel systems and indicates how much computation work is needed, as the number of processors increase, to maintain a constant efficiency. To formally express the isoefficiency metric it is necessary to define a related function called the total overhead function.

2.3.3.1 Total Overhead Function

The total overhead function is the total sum of the overhead time incurred due to parallel processing by all the processors. It includes communication costs, idle time, synchronization and sequential fraction of the algorithm.The overhead function is expressed as,

$$\begin{aligned} T_o(p, T_s) &= pT_p - T_s \\ T_p &= \tfrac{T_o(p,T_s)+T_s}{p}, \end{aligned} \tag{2.4}$$

in order to maintain constant efficiency we express it in terms of the overhead function

$$E = \frac{1}{1 + \frac{T_o(p,T_s)}{T_s}} \tag{2.5}$$

As described in the literature [34], problem size is defined in terms of the number of operations performed by the best sequential algorithm. If we consider that in a serial processor there is a linear relationship between time and operations, then we can represent the problem size, called W, by T_s. Reformulating Equation 2.5, we obtain

$$E = \frac{1}{1 + \frac{T_o(p,W)}{W}} \tag{2.6}$$

meaning that, if W is increased while the number of processors is fixed, the efficiency will increase due to the slowdown that the overhead function has over the problem size. In contrast, if the number of processors increase while the problem size is maintained fixed, the overhead function will increase due to overhead processing and, as a consequence, the efficiency will drop [35]. Efficiency can be maintained fixed if the ratio of $\frac{T_o}{W}$ in Equation 2.6 is kept constant. Therefore, for a fixed value of efficiency we have

$$W = \frac{E}{1 - E} T_o(p, W) \tag{2.7}$$

Let K be the constant representing the efficiency to be maintained, then Equation 2.7 can be rewritten as,

$$\begin{aligned} K &= \tfrac{E}{1-E} \qquad \text{constant} \\ W &= K T_0(p, W), \end{aligned} \tag{2.8}$$

where the W is a function dependent on the number of processors p and can usually be obtained from Equation 2.8 by algebraic manipulations [35]. The isoefficiency function is a function that relates work to number of processors and must satisfy the relation $W = \theta(f(p))$.

2.3.4 Optimal Effectiveness Metric

The optimal effectiveness metric is a scalability metric that measures the number of processors for which the parallel system can achieve optimal effectiveness as work increase [52]. It describes the effectiveness of a system as a ratio of performance to cost. Performance is represented as the computation rate and cost as the resources employed to perform a task. The optimal effectiveness metric is expressed as

$$\Gamma_{opt} = \overset{max}{p} \left\{ \frac{W}{pT_p^2} \right\} \tag{2.9}$$

2.3.5 Cost Metrics

The definition of cost for the parallel community is defined as the sum of the time spent for each processor to solve a problem. Mathematically speaking, cost is the product of the parallel execution time and the number of processors (pT_p)[34]. Other definitions related to cost involve the relation of analyzing systems that are cost-effective. In such situations it is necessary to consider the tradeoffs between the speed and the number of processors to use under a given budget. The study of cost in parallel systems is complex due to the many different factors involved in the analysis [49], although, there has been some preliminary work and related metrics of this study to evaluate parallel systems. Some related cost metrics are:

Costup metric: Presented by Wodd and Hill [85], this metric expresses the ratio of the cost of a parallel system to the cost of a serial system. It determines the cost effectiveness of a parallel system if the costup exceeds speedup.

Profitup metric: Presented by Yan and Zhang [86], this metric expresses the ratio of the net profit obtained by a parallel system to the net profit obtained by a serial system. The net profit of a system is expressed as the difference between the production function and the cost of the system. The production function is the profit gained from the performance of the system and is expressed in the context of economics [76]. The profitup is defined as

$$\begin{aligned} Profitup &= \tfrac{Profit(1)}{Profit(p)} \\ Profit(p) &= Pro(p) - Cost(p) \end{aligned} \tag{2.10}$$

2.3.6 Other metrics

Most applications are affected by two characteristics:

Availability: The capacity of a request being served at the moment it is been placed. It measures turn around time, throughput, etc. Some metrics that follow this characteristic are:

- Transaction Per Minute (TPM) uses a software testing tool that generates transactions against the system and measures the results. It depends on the system and the transaction, estimations such as the number of users the system can handle can be done by this metric.
- Bandwidth measurement is the amount of data that can be transmitted over the channel per unit time. It is represented in percentage, *e.g.* % CPU, % Network utilization, % Memory utilization, etc. It helps to determine bottlenecks.

Responsiveness: The speed at which the request is satisfied. It is frequently measured in floating points-per-second (Flops) or millions of instructions per second (MIPS). Some metrics that follow this characteristic are:

- Response Time is the measure of time that it takes from the moment a transaction is requested until the moment it is satisfied by the system. It is a helpful tool when establishing the adequate response time for a given transaction doing some type of survey to the users.
- Retrieval Time is the measures of time that it takes for a system to deliver all the results once its starts to deliver the first result. It expresses download time.
- Transaction Time is the sum of the response time and retrieval time.
- Network Time is the measures of time that the data spends on the communicating channels. The respond time and retrieval time depend on- and include- network time.

2.4 Economic Models for Performance Analysis

Previously we discussed the evolution that high performance computing is having in commercial applications, and that customers and vendors are now concerned about the relation between cost, profit and performance. The rapid advances of commodity processors and network technology have made possible the creation of custom-designed platforms such as the Beowulf Project [6], which can generate a better price/performance ratio than some of the most robust parallel systems from experienced and traditional parallel manufacturers.

The task for changing and evaluating IT infrastructure is not easy. The risks are very high, and many other factors have to be taken into consideration when the range of possibilities vary from very expensive, proprietary, robust systems to custom-design massive parallel, commodity systems. In summary, the evaluation of parallel systems based on the

measure of its speed is no longer sufficient, and there are more elements that should be studied before reaching a decision.

Some economic models have been used to represent the financial impact of computer systems. In other words, they represent the performance (in dollars) of the computer systems. Many of these models have evolved to predict the costs of managing and operating a distributed computing environment. The analysis of these results is oriented to minimize these costs. There exist two common cost models that have been widely used for the evaluation of computer systems, the Total Cost of Ownership (TCO)[11] and the Total Economic Impact(TEI)[20].

2.4.1 Total Cost of Ownership (TCO)

Total Cost of Ownership is a model that measures all the costs (acquisition, installation, usage, maintenance, changing, etc.) associated with an information technology project, as shown Figure 2.4. The Gartner Group [5] is a consultant IT organization that has been the largest and most vocal proponent of the TCO concept [11]. Applying this model is a starting point for many IT managers to understand certain factors that affect the management and operation of computing systems and to balance these factors with business benefits. The TCO model is composed of several costs that can be categorized as:

In-budget costs: In-budget costs are captured by capital and labor costs of the IT infrastructure. Capital costs include all the hardware and software, communication equip-

[5] `http://www.gartnergroup.com/`

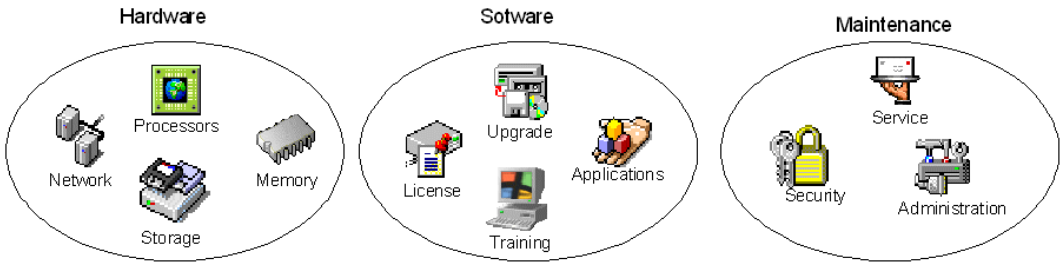

Figure 2.4 The Total Cost of Ownership (TCO) model

ment, additional space, etc. Labor costs correspond to professional staff or outsource contracts for computing services, such as maintenance. These costs are tangible and are the most simple to identify.

Off-budget costs: Off-budget costs are very difficult to quantify, as they measure the financial impact of the technology upon the organization it serves. Off-budget costs are broken into three categories: productivity impact, end-user IT and revenue impact. The first relates to the cost of the time that end-users spend in performing activities related to technology but not related to its primary function, such as surfing on the web. The second relates directly to the costs of not using the technology, such as coffee break or going to the bathroom. The third and last category has a great impact on cost; it measures the cost of the time that end-users could not perform their job because of technology inefficiencies and failures. It measures the downtime and recovery costs of the system.

2.4.2 Total Economic Impact (TEI)

Total Economic Impact is an extension of the basic cost model created by the Giga Information Group [6] in 1997. It is a value-based methodology for quantification of costs, benefits and risks in the IT environment, as illustrated Figure 2.5.

Figure 2.5 The Total Economic Impact (TEI) model

TEI is defined as the net total benefits (cost savings + business benefits + flexibility - total costs) divided by the total costs. It is a more comprehensive measure than other cost/benefit models. It considers four elements of any initiative:

- Benefits, which represent the value delivered to the business by the proposed IT infrastructure. They are measured in a period of time and have a direct impact in the business profit.
- Costs, relate directly to the TCO model. This element represents the investment necessary to capture the value, or benefits, of the proposed IT infrastructure.
- Flexibility, represents the additional capacity invested for future business benefits that cannot be deployed at the moment. The adaptation of software developers to

[6] http://www.gartnergroup.com/

generate new applications that exploit the recent acquisition of the high speed interconnection in the department, would be an example of such future benefit.

- Risks, are difficult to calculate. Since the future cannot be accurately predicted, there is risk inherent in any project. It involves factors like:
  - Products, which might not deliver the functionality expected.
  - Vendors, which might be changed or replaced.
  - Architecture, which might not support the new changes.
  - Culture, which involves the risk that the organization will not absorb the new technology or adapt to its implementation.

In the early years of HPC, no single methodology for return on investment(ROI) analysis was dominant because only large enterprises could afford and justify the expense of computing. Therefore, the market for standardized services and methods of cost measurement was small. In a time moving more and more at “Internet speed”, businesses are faced with increasing challenges to rapidly and accurately estimate the value of their IT investments. Recently, methods have been developed to apply these models to actual computing environments and utilize the output as a guide for improving processes, skills and technology. The end result of these improvements is a better return on an organization’s technology investments.

CHAPTER III

COST/PROFIT EFFECTIVE METRIC

In this chapter, we introduce a framework for analyzing the factors of cost and profit that affects the decision of replacing or upgrading a parallel system. We will start by explaining some of the challenges that IT managers have to handle in the IT environment. Then, we will continue to explain the development of this framework and conclude with the introduction to the definition of the cost/profit effective metric.

Nowadays, an IT department has many challenges. Advances in technology have generated a proliferation of data centers full of heterogeneous systems, from workstations to clusters, mainframes, MPPs, SMPs, etc. An IT manager has the responsibility to administrate all these resources in an efficient manner, trying to fulfill the users (customers) needs, and to meet the business goals. To have a reliable, powerful and at-the-edge-of-technology infrastructure, IT managers need to rely on a very broad and sometimes confusing technology market. With a set of users that demand services in order to meet the business goals, a limited budget for expanding or maintaining the IT infrastructure, and a group of vendors willing to offer the best price/performance products, the following questions arise for the IT managers:

- What is the most appropriate technology that satisfies the business goals?
- When is it convenient to replace or upgrade the parallel system infrastructure?

- Is the productivity of the parallel system justifying its investment?
- How to calculate the return on investment of a parallel system based on its performance?

These are important questions for an IT manager. The IT department now plays an important role in many organizations. IT managers are now required to present their projects in a language that upper management executives understand. This language is based on cost and profit.

To help IT managers with this task, we develop a framework that summarizes the most relevant factors of performance and economics when evaluating parallel systems. The main aspects of this framework are the following:

- The definition of parallel system effectiveness based on performance, cost and profit.
- The identification of relevant factors that are involved when deciding what parallel system to acquire.
- The analysis of the two scalability approaches in parallel systems: scale-up and scale-out.
- The depreciation factor in parallel computing.
- The definition of the cost/profit effective metric as a measurement of performance, cost and profit to evaluate parallel systems.

The following sections describe these aspects in detail.

3.1 Defining Parallel System Effectiveness

In section 2.3.2 of chapter II we introduced the definition of efficiency based on the performance of a parallel system. Effectiveness is related to efficiency. The effectiveness of

a parallel system can be defined as the ability of the system to perform efficiently under given circumstances [85]. These circumstances can be presented as workload, time, or even cost. For example, the cost-effectiveness of a parallel system can be measured by the ratio of its performance to cost. If the performance achieved by the system is greater than the cost, the system is cost-effective [86]. Moreover, let us compare the cost-effectiveness of two systems, *e.g* S_1 and S_2, then S_1 is more cost-effective than S_2 if the ratio of performance to cost of S_1 is greater than the ratio of performance to cost of S_2. Equation 3.1 represents this example, $\Gamma_{cost}(S)$ represents the cost-effectiveness of a system S.

$$\Gamma_{cost}(S) = \frac{Performance}{Cost}$$

$$\Gamma_{cost}(S_1) = \frac{P_1}{C_1} > \frac{P_2}{C_2} = \Gamma_{cost}(S_2) \qquad \begin{aligned} P_1 &= \text{Performance of } S_1 \\ P_2 &= \text{Performance of } S_2 \\ C_1 &= \text{Cost of } S_1 \\ C_2 &= \text{Cost of } S_2 \end{aligned} \tag{3.1}$$

From this example, we then define the effectiveness of a parallel system in terms of performance, cost and profit. The cost/profit effectiveness of a parallel system can be measured by the ratio of the profit based on the productivity of the system performance in a period of time, to the total cost spent operating and maintaining the system over the same period of time. Then, we can state that a parallel system is cost/profit effective if the profit obtained from the system performance is greater than the total cost of operation of the system in a given period of time. This definition will be formalized by the cost/profit effective metric as a measure of performance, cost and profit of a parallel system.

3.2 Identification of Factors

When evaluating a parallel system, there are many important factors besides its purchase cost and its raw performance. We identify some of these factors and summarized them in the following four categories:

Potential: Potential factors are those that influence the system's capability to efficiently perform a service. Examples include, performance of the system, the installation of the system, network interconnection requirements, porting applications, and compatibility with other systems.

Financial: Financial factors directly impact the budget plan assigned to the system. Analyses of total cost of ownership (TCO) are commonly used to represent these factors. Some financial factors are maintenance cost, upgrades, training, etc.

Environmental: Environmental factors are those that represent the IT environment where the system is going to perform. For example cooling power, electricity, floor space, etc.

Cultural: Cultural factors are the most difficult factors to quantify. They correspond to the cultural formation that the organization has developed around the system, like, rejection of the use of new technology such as operating systems, software, applications, etc.

These categories are not exclusive, there exists a close relationship among their factors as we can see from Figure 3.1. For example, the acquisition of new nodes to a cluster system (scale-out approach), will impact several factors like the budget plan (financial fac-

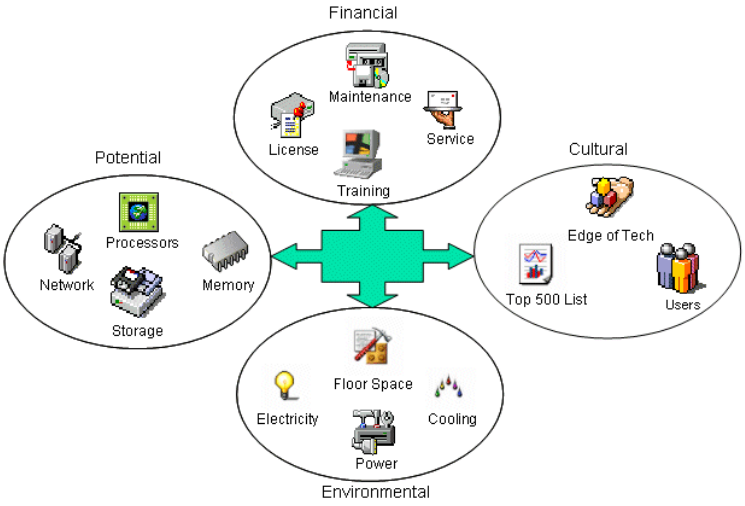

Figure 3.1 Categories that are considered when evaluating parallel systems

tors), delay of some tasks because of the time spent on installing the nodes in the cluster (potential factors), arranging the space to accommodate the nodes in the data center (environmental factors), and users adaptation to exploit the new system (cultural factors). All these factors should be listed and considered to make the right evaluation among parallel systems.

3.3 Scalability in Parallel Systems

Scalability is a very important issue to take under consideration when evaluating the expanding capabilities of a parallel system. How much can the system grow? Is a question that depends on the physical limitations of the system and the economic limitations of the organization that is investing in the system. As we mentioned in section 2.2, scalability analysis have focused mostly on performance. However, there are other implications be-

sides performance that should be studied. To analyze in what direction a particular system should grow, we have to study two approaches: scale-up and scale-out.

Scale-up: Scale-up is seen as a vertical way to scale [51]. The system can be expanded by upgrading some of its components like CPU speed (going from slower to faster processors), adding more memory, having faster access to the network, etc. Any changes that are performed inside the hardware components of the system, such as enhancing its performance, can be considered as scaling-up [64]. From a hardware point of view, a parallel system can scale-up to a certain limit. After that limit is reached other choices should be considered. Parallel system architectures, where the scale-up approach is commonly used, are in traditional HPC systems built in one box (SMP, NUMA, etc.).

Scale-out: Scale-out is the other approach that we can choose when scaling a parallel system. In contrast to the scale-up approach, the scale-out approach adds more elements (nodes) to the same system to obtain power. In principle, there are no limits for scaling-out, although, there are several considerations in system software, network components and environmental factors [64, 51]. Parallel system architectures, where the scale-out approach is commonly used, are distributed HPC systems (MPP, Cluster, etc.).

Not all systems scale-up or -out properly. They depend on the operating system and the applications used. The scale-up approach tends to be less concerned about software architecture. In contrast, the scale-out approach depends on software design so the system can be deployed properly and effectively. Some systems will scale-down (meaning that

the performance has suffered a slow down), although resources have been added, causing resources to be wasted and incrementing the costs in the organization.

3.4 The Depreciation Factor in Parallel Computing

Depreciation is another important factor that should be considered when making the decision to replace or upgrade a parallel system. Depreciation is defined as the loss value in time of an asset [1]. Parallel computers, like any other computer equipment, are considered an asset within any organization. Calculating the depreciation of an asset can be complicated for many cases. In our case we will consider the depreciation factor of a computing system as the cost of the system divided by its useful time operating in the organization. This calculation is called the straight-line depreciation. In general, the depreciation factor of a computer system (from workstations to supercomputers) is estimated, in accounting [67], for three years. Some argue [67, 42] that this estimate does not considers the rapid growth of information technology systems over the years. Moore's Law [57] is an example of this.

Probably the most known law in IT is Moore's Law. Originally proposed in 1965, by Gordon Moore, the co-founder of Intel Corporation [46]. Moore's observation was based on chip transistors, stating that every eighteen months the number of transistors that can be accommodated in a chip will double [57]. This law has had a direct effect on the processors speed and, therefore, on computing performance. For over 30 years, Moore's Law has been taken under consideration for price/performance analyses, although the growth has

slowed down (between 18 to 24 months), it constantly transforms the IT environment by permitting commodity processors to achieve high performance similar to the earliest Cray systems.

The IT market has followed Moore's Law by incrementing exponentially the computational performance as cost has been kept constant. On the other hand, the cost of fixed computational performance declines exponentially. In summary, Moore's Law predicts that every 18 to 24 months the price of computation reduces to 50%. As we can notice, this depreciation is much faster than the three years depreciation on computer systems for accounting. As a consequence, a parallel system should be fast and scalable to stay ahead of the best workstation, whose speed doubles every 18 months [40].

3.5 The Cost/Profit Effective Metric

In the creation of any metric it is important to state what is that we would like to measure, and what impact the metric will have when analyzing the results. The parameters need to be qualitative and quantitative enough, giving a good representation of the metric.

The cost/profit effective metric is represented in monetary units, based on the performance of the system during a period of time. The results can lead to analyses that support the decision of acquiring more resources or moving to a new infrastructure to enhance the business.

The parameters that constitute the cost/profit effective metric are a representation of parameters used on performance metrics and parameters used on economic models. These

parameters can be grouped, according to our previous categorization, by potential and financial factors. From the potential factors we have parameters like speedup, throughput, parallel execution time, sequential execution time, number of processors, and efficiency. From the financial factors we have parameters such as cost, production function, and profit.

The profit-effective metric [86], in conjunction with the cost-effective metric [85] discussed in section 2.3, were used to define the parameters for the cost/profit effective metric. From the cost-effectiveness concept, performance must be greater than cost for the system to be cost-effective. Thus, as discussed previously, we establish that if the performance of a parallel system directly affects the profit of the business, then the parallel system is cost/profit effective if the profit obtained by its performance is greater then its total cost. This statement is represented as follows:

$$\begin{aligned} \Gamma_{profit(S_p,t)} &= \text{cost/profit effective metric} \\ \Gamma_{profit(S_p,t)} &= \tfrac{Profit(S_p,t)}{Cost(S_p,t)} \end{aligned} \tag{3.2}$$

As we can observe from Equation 3.2, the cost/profit effective metric is composed of two functions, the $Profit(S_p, t)$ and $Cost(S_p, t)$. The $Cost(S_p, t)$ function determines the total cost incurred by acquiring and using the system with a configuration S_p with p processor/nodes performing in time t. The $Profit(S_p, t)$ function represents the net profit of the system with a configuration of S_p with p processor/nodes performing in time t. The $Profit(S_p, t)$ function is defined as follows:

$$Profit(S_p, t) = Pro(P(S_p), t) - Cost(S_p, t) \tag{3.3}$$

substituting in Equation (3.2), we have:

$$\Gamma_{profit}(S_p, t) = \frac{Pro(P(S_p), t)}{Cost(S_p, t)} - 1 \tag{3.4}$$

where we will call $\frac{Pro(P(S_p),t)}{Cost(S_p,t)}$ as the cost/profit effective factor (Γ_{factor}). This factor determines the effectiveness of the system in terms of profit and cost.

As an example, let us consider the scenario where a new project has just begun and there is an acquisition for a new parallel system. At the beginning of the project we expect no productivity because the system has just been acquired, this is represented by $Pro(P(S_p), 0) = 0$. Hence, the result of the cost/profit effective metric in $t = 0$ is a negative value. This type of behavior is expected since the only values present at $t = 0$ will be the purchase cost of the parallel system, given by the $Cost(S_p, t)$ function. As time advances, we would expect $Pro(P(S_p), t)$ to be less than $Pro(P(S_p), t + 1)$ and $Cost(S_p, t)$ to be less than or equal to $Cost(S_p, t + 1)$, depending on the maintenance and other factors that alter the cost of ownership. For a parallel system to be cost/profit effective the $Pro(P(S_p), t)$ function should reach the point where it grows faster than the $Cost(S_p, t)$ function in time, as represented by Figure 3.2.

We can also see from Figure 3.2 that in the acquisition of any new system, there exists an adaptation gap before the $Pro(P(S_p), t)$ function can outgrow the $Cost(S_p, t)$ function. As shown in Figure 3.2, there is a point in time where these two functions meet. After this time the business starts obtaining the return of investing in the system. The main idea, to accelerate the return, is to keep the adaptation gap small as possible. The generation

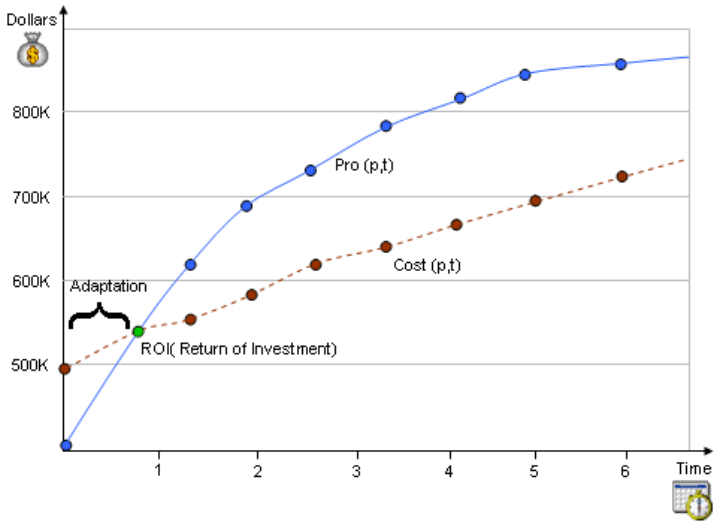

Figure 3.2 The behavior of the profit-effective metric in the evaluation of a parallel system

of small adaptation gaps can be accomplished by creating a plan beforehand that covers the installation, migration of applications and training, staff and users, of the system. The following chapters define in detail the $Pro(P(S_p), t)$ and the $Cost(S_p, t)$ functions. We describe both functions and discuss their applications in terms of evaluating parallel systems.

CHAPTER IV

THE PROFIT FUNCTION

Before getting into any formal definition of the profit function ($Pro(P(S_p), t)$), we will present a correlation of this function with some economic definitions. In economics, a production function is defined as a function that shows how much output is produced when specific amounts of each factor of production are being used [76]. A production function is commonly defined as $y = f(L, K)$ where L represents labor and K, capital. In our case, the production function we want to define is described by two factors of production: the performance of the system ($P(S_p)$), with configuration S_p and p processor/nodes, and the time (t) that the system has been in production. The output produced is the business profit, represented in monetary unit, obtained from the performance of the system during that time, as shown in Figure 4.1.

We will illustrate this definition of production function in Table 4.1, which shows the values of a production function f, during time t, of a parallel system S configured for p processors. We use the relationship of $speedup(p)$ and p, used in the work of Yan and Zhang [86], as $speedup(p) = f_s p$, where f_s is the speedup factor assumed to be $f_s = 0.25$ [86]. We assume that the speedup of system S follows Amdahl's Law, that is, the speedup

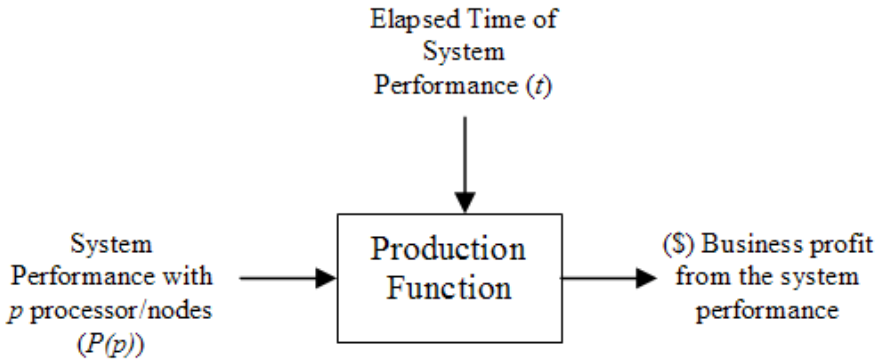

Figure 4.1 The cost/profit effective production function

of system S will be incremented as the number of processors p are increased and the problem size is fixed. This is observed in Table 4.1.

The unit gain rate ($\varphi(P(S_i))$), showed in Table 4.1, represents the gain in dollars from the performance P of the system with configuration S_i in unitary time. In Table 4.1 we can see that system S_1 (with four processors and a $speedup(4)$ of 1.0) has a unit gain rate of \$500[1] meaning that this was the profit obtained by the performance of S_1 in one unit time. We can also see that system S_2 has a unit gain rate of \$1,000, twice the unit gain rate of S_1, reflecting an enhancement in the effectiveness of the system with eight processors ($speedup(8) = 2.0$) compared with the configuration of S_1 ($speedup(4) = 1.0$) with four processors.

As time advances, the unit gain rate accumulates the productivity of the system performance during that time. For example, from the Table 4.1 we can observe that in time

[1]Assuming a system that generates 10 transactions per second at a value of \$0.014 per transaction, taken from a study done by the Standish Group International in 2002 (http://www.standishgroup.com)

Table 4.1 Production function for a parallel system with S_i configurations

			Production function f presented in thousands of $									
System Configuration (p)	$speedup(p)$ **(f_s=0.25)**	**Unit Gain Rate ($\varphi(P)$)**	t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	t=9	t=10
S_0		0	0	0	0	0	0	0	0	0	0	0
$S_1(4)$	1.0	0.5	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
$S_2(8)$	2.0	1	1	2	3	4	5	6	7	8	9	10
$S_3(16)$	4.0	2	2	4	6	8	10	12	14	16	18	20
$S_4(32)$	8.0	4	4	8	12	16	20	24	28	32	36	40
$S_5(48)$	12.0	6	6	12	18	24	30	36	42	48	54	60
$S_6(64)$	16.0	8	8	16	24	32	40	48	56	64	72	80
$S_7(128)$	32.0	16	16	32	48	64	80	96	112	128	144	160
$S_8(256)$	64.0	32	32	64	96	128	160	192	224	256	288	320
$S_9(512)$	128.0	32	32	64	96	128	160	192	224	256	288	320
$S_{10}(1024)$	256.0	32	32	64	96	128	160	192	224	256	288	320

$t = 2$, the system performance of S_1 has a productivity of $1,000, which is twice the productivity of the system in $t = 1$. For S_2, the productivity is $2,000, twice $t = 2$, and so on. Therefore, the production function f will behave linearly with respect to time (t) when the unit gain rate of the system is maintained fixed. The graph in Figure 4.2 shows the partial production function when $\varphi(P(S_i))$ is fixed.

On the other hand, if time is fixed, we cannot assure a linear behavior for the production function, mainly because the unit gain rate $\varphi(P(S_i))$ will vary according to the system performance. In the example of Table 4.1 we assumed that the unit gain rate behaves constant for the last three configurations of the system, that is, the system will not increase its productivity more than the productivity obtained by the performance of the system S_8. From this point on, the productivity of the system has reached its limit by having a constant behavior. The configurations S_9 and S_{10} will have no additional productivity to the system effectiveness. The graph in Figure 4.3 shows the partial production function when time t is fixed.

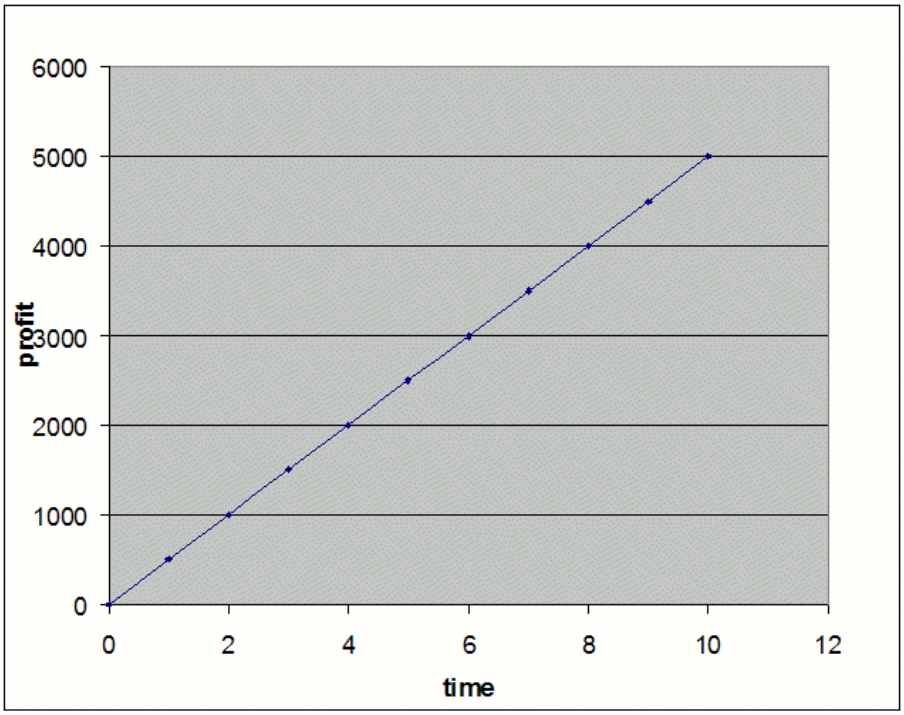

Figure 4.2 Partial production function when $\varphi(S_i)$ is fixed

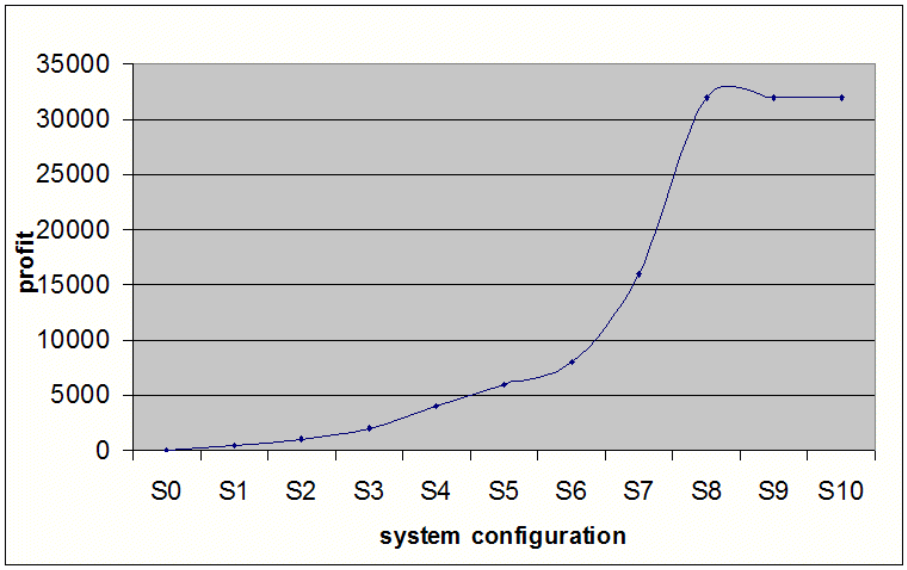

Figure 4.3 Partial production function when t is fixed

In Table 4.1, the 128 additional processors in configuration S_9 will not influence an increase on the system productivity. The only increment that will reflect the adding of 128 processors is the cost of the system, which we will discuss in Chapter V.

The example of Table 4.1 represents an "ideal" scalable system, where there exists an increment in the performance of the system while resources are increased. The production function f as described in Table 4.1 also follows a law in economics called the "law of diminishing returns" [54]. This law states the following:

> *When one of the factors of production is held fixed in supply, successive additions of the other factors will lead to an increase in returns up to a point, but beyond this point returns will diminish.*

In other words, small increments on one of the inputs of the production factors, while holding others constant, results eventually in smaller and smaller increases in the added output of the production function. The law of diminishing returns does not take effect immediately in all production functions, but eventually increasing returns will stop and decreasing returns will set in [70]. The graph in Figure 4.4 represents the behavior of the production function for this system. We can observe that the production function behaves linearly in time (t), but it holds a different behavior for $\varphi(P(S_i))$.

We understand that not all systems behave like the example described above (a system with a scale-out approach, will have different behavior) and that many of the factors are dependent on the workload or problem size, system configuration and business model for which the productivity is being measured. Before we discuss these details, however, we wanted to give the basis for defining the production function $Pro(P(S_p), t)$.

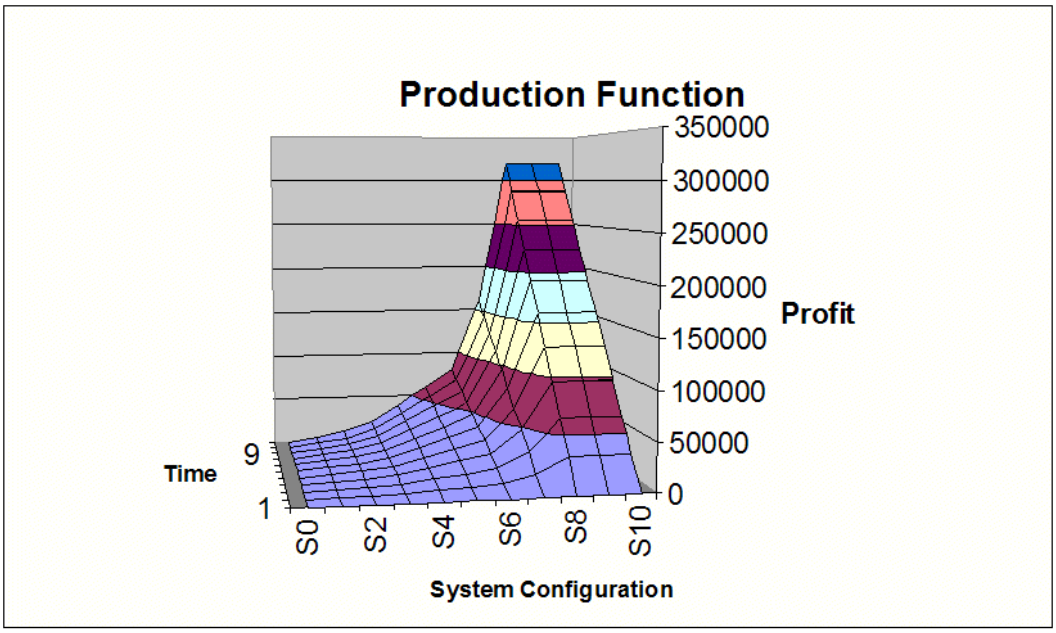

Figure 4.4 Production function for Table 4.1

Let $\Delta\varphi(P(S_i))$ be the difference of $\varphi(P(S_i))$ and $\varphi(P(S_1))$ for any $S_i \in S$ and let, S_1, be the actual system configuration. Then we can define the production function $Pro(P(S), t)$ as:

$$Pro(S_i, t) = \begin{cases} \varphi(P(S_i)) & \text{for } t = 1 \\ (\varphi(P(S_1)) + \Delta\varphi(P(S_i)) \times t & \text{for } t > 1 \end{cases} \tag{4.1}$$

with time t and unit gain rate $\varphi(P(S_p))$ as production factors. In general, we are looking for a production function that will behave as follows:

1. $y = f(\varphi(P(S_p)), t)$ will behave linearly in time t if for any two production values $y_i = f(\varphi(P(S_k)), t_i)$ and $y_j = f(\varphi(P(S_k)), t_j)$, such that $0 < t_i < t_j$ and K constant, then $y_i < y_j$.

2. For $y_i = f(\varphi(P(S_i)), t_k)$ and $y_j = f(\varphi(P(S_j)), t_k)$ such that $S_i < S_j$ and K constant, we have that $y = f(\varphi(P(S)), t_k)$ behaves, with respect to the unit gain rate, as follows:
   (a) constant ($y_i = y_j$) if $f(P(S_i)) = f(P(S_j))$
   (b) sublinear ($y_i > y_j$) if $f(P(S_i)) > f(P(S_j))$
   (c) linear ($y_i < y_j$) if $f(P(S_i)) < f(P(S_j))$

In our example of Table 4.1 we observed how the production function f behaved linearly with respect to time, linearly for the unit gain rate $f(P(S_{1-8}))$ and constant for $f(P(S_{8-10}))$.

Let us now view another important economic concept that is interesting to relate to this analysis, marginal productivity [56]. Recalling the production function $y = f(L, K)$, the marginal productivity of labor (L) represents the additional output as a result of adding one unit of labor with all other inputs fixed. In our case, let us relate performance to labor. Then, we define the marginal productivity of performance as the additional profit gain from increasing the performance of the system with a given configuration S_i during a fixed time t. In our terms, we will define marginal productivity of performance as:

$$M_{S_i} = \frac{\Delta\varphi(P(S_i))}{\Delta(P(S_i))} \tag{4.2}$$

As an example, let us considered S_1 from Table 4.1. Let us assume that the unit gain rate of \$500 is obtained from 72[2] units of work produced by S_1 in one hour. From Table 4.1, S_2 (system with eight processors) has twice the $speedup(p)$ of S_1, as a consequence, we assume it produces twice the units of work (144 units). The unit gain rate

[2]Assuming the parallel system generates 1.2 operations per minute

obtained for S_2 (according to Table 4.1) is \$1,000. Then, we can calculate the marginal productivity based on the performance of the S_2 configuration as:

$$\begin{aligned} M_{S_2} &= \frac{\Delta\varphi(P(S_2))}{\Delta P(S_2)} \\ M_{S_2} &= \frac{\$1000-\$500}{144-72} \\ M_{S_2} &= \$6.94 \end{aligned} \tag{4.3}$$

We can interpret this result as follows: over the range of performance of a system with four to eight processors each additional performance unit adds approximately seven units to the profit gain. The calculation of the marginal productivity can be helpful to identify how much profit gain is obtained per unit of work when the performance is incremented.

4.1 Expressing $\varphi(P(S_i))$ in terms of $P(S_i)$

The cost/profit effective metric is a measure that represents the relationship between business and technological factors for evaluating high performance computing systems. Therefore, each of the parameters that compose this metric should represent this relationship. As mentioned before, $P(S)$[3] describes the measure of the system performance with configuration S and $\varphi(P(S))$ represents the gain rate that $P(S)$ has produced over unitary time. Now, the question is: "how can we obtain the unit gain rate from system performance?" Many IT executives find this question difficult to address when they need to justify their projects to upper management executives, whose main focus is on business gain [53].

[3]We will substitute the notation S_p for just S in order to simplify notation, meaning that S will be implicitly dependent on the number of processors/nodes p for the system

There exist many measurements for system performance [60]. Some measures focus on the resources of the system (% CPU utilization, % network latency, etc.), others, on the throughput (jobs executed, response time, internal transactions, etc.). In our case, we are interested in the representation of system performance, as Norton describes in his work [60], by a measure of end-to-end response time. That is, system performance is measured by the impact it has over business objectives to understand how much profit or benefit the system is bringing to the organization. The end-to-end response time measure is observed from a business point of view which focuses interest on service, capacity planning and evaluation of systems.

With the complexity of many systems and their applications, it is difficult to know what to measure and what should represent the performance of the system. Therefore, parallel system performance is determined on a case-by-case basis dependent on the application, process or service the system is used for. For example, a travel agency that sells tickets online is interested in the amount of transactions per second generated by its parallel system while an engineering computing center is interested in the turnaround times generated by its parallel system for modeling and simulation projects. For this reason, we will generalize the unit gain rate by expressing it in terms of the impact that the performance model $P(S)$ has over the business.

The unit gain rate $\varphi(P(S))$ is expressed as the value, in monetary units, of the work completed by the performance model $P(S)$ in unitary time. As an example, let us consider a business model which focuses on transaction processing. If the parallel system used by

this model generates ten transactions per second (TPS), and each transaction has a value of $0.014, then the unit gain rate obtained from the performance model is expressed as,

$$\begin{aligned} &= 0.014 \frac{\$}{trans.} \times 10 \frac{trans.}{sec.} \\ &= 0.14 \frac{\$}{sec.} \end{aligned} \tag{4.4}$$

Let V be the value for each unit of work performed by the system, and $P(S)$ the performance model by which the system is measured. Then, for this business model the unit gain can be expressed formally as,

$$\varphi(P(S)) = V \times P(S) \tag{4.5}$$

We can observe in the graph shown by Figure 4.5, that for this business model the unit gain rate $\varphi(P(S))$ will behave linearly as $P(S)$ increases. Moreover, as the performance of the system increases, the performance model $P(S)$ will also increase by generating more transactions per second, and as result $\varphi(P(S))$ will also increase. Therefore, we can state that the performance by which the system is measured will depend on the business model. In the example above, the business focus on transaction processing, therefore TPS represents the measurement for the performance model by which the system affects the business.

To have a better understanding of how to express the unit gain rate in terms of performance, let us describe another example based on a business model centered on production

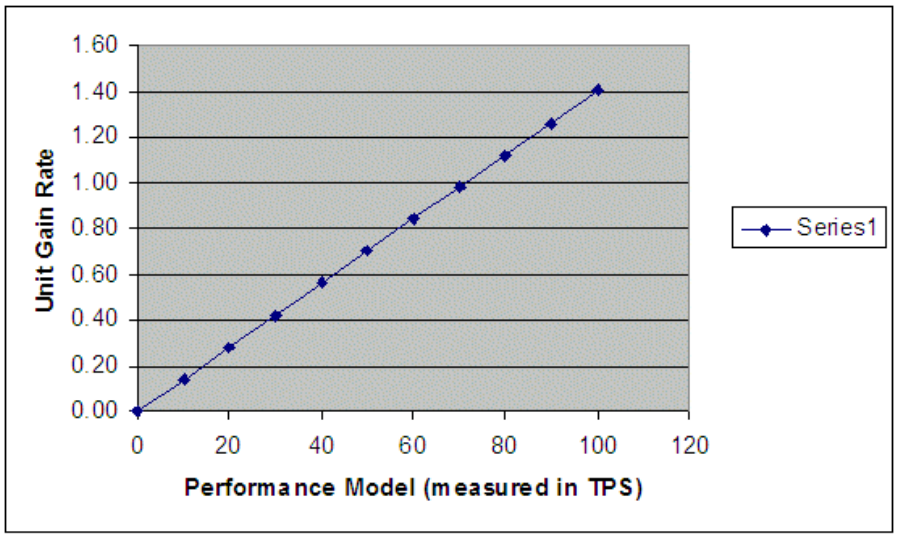

Figure 4.5 $\varphi(P(S))$ expressed on transaction processing model

cycles. For this business model the performance is measured by turnaround times. The idea is to reduce turnaround times to accelerate the generation of a product. Production cycles may need several runs of the same job to assure quality, and each is composed of several tasks, such as design, simulation, testing, etc. Then, for this business model we present the total time of a product cycle as the product of the number of jobs run per cycle and the total time spent in the computational and non-computational tasks of a job. The computational tasks are based on the system performance, expressed as the average time elapsed to execute a job ($\frac{time}{job}$). The non-computational tasks correspond to the time spent on processes that do not require computing systems (*i.e.* design of the product). Then, if we have a product cycle of 10 days, where 5 days correspond to the design of the product

and simulations take 5 days in a parallel system, assuming that only one run of the cycle is necessary and the value of the product is $20. Then, we can express the unit gain rate as,

$$\begin{aligned} &= 20\frac{\$}{product} \times \frac{1product}{(5\frac{day}{job}+5\frac{day}{job})1job} \\ &= 2\frac{\$}{day} \end{aligned} \tag{4.6}$$

If the performance of the system $P(S)$ is improved from 5 to 2 days then the unit gain rate will increase:

$$\begin{aligned} &= 20\frac{\$}{product} \times \frac{1product}{(5\frac{day}{job}+2\frac{day}{job})1job} \\ &= 2.8\frac{\$}{day} \end{aligned} \tag{4.7}$$

This is illustrated by the graph in Figure 4.6 where the improvements on system performance $P(S)$ will increase the unit gain rate. We can also observe from this graph that the unit gain rate is bounded by the non-computational tasks, that in this example take 5 days. Moreover, as the simulations performed by the parallel system take less time, the unit gain rate will increase up to a maximum of $4\frac{\$}{day}$ as shown in Figure 4.6.

Finally we express formally the unit gain rate for this business model as,

$$\varphi(P(S)) = \frac{V}{N(t_{nc} + \frac{1}{P(S)})}, \tag{4.8}$$

where V is the value of the product, N is the number of times the job is executed on each product cycle, t_{nc} represents the time spent on non-computational tasks and $P(S)$

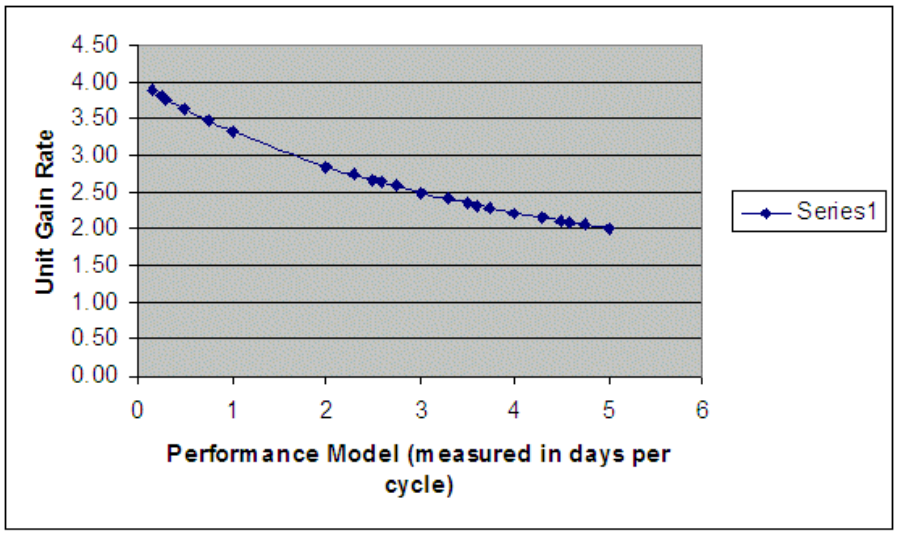

Figure 4.6 $\varphi(P(S))$ expressed on production cycles model

represents the performance ($\frac{jobs}{time}$) that the systems spends on the execution of a job. It is worth noting that the behavior of the unit gain rate for this business model is similar to the behavior of a scalar parallel system that follows Amdahl's Law. As mentioned earlier, for this example the $\varphi(P(S))$ presents an upper bound based on the non-computational tasks (O($\frac{V}{Nt_{nc}}$)). It is also important to notice that if the majority of time of a product cycle is assigned to non-computational tasks, then any enhancement on the system performance will not have a great impact on the overall product cycle.

In spite of the diversity of performance models for parallel system, we found interesting to generalize them in three paths [63], speed-up (to reduce turnaround time), throughput (to solve more problems or serve more users), and/or scale-up (to solve bigger problems). These paths are not exclusive and will sometimes overlap according to the objectives of the system and the impact on the business. In the following sections we discuss

the performance measurements used in these paths and give some examples of business models that fall into each case.

4.1.1 Performance Based on Speed-up (Turnaround Time)

Speed is one of the reasons for parallel computing. A way to measure speed is by calculating the turnaround time of the system. The term "turnaround" is commonly used to represent the time it takes for a job to be executed. Turnaround time is also called end-to-end delay [81]. The improvement on turnaround times increases productivity. As a consequence, an increment on productivity has a direct impact on profit and business goals. Turnaround time is defined as the interval between the time that the system has received a job and the time that job is executed. This time might include the time that the job spends in the queuing system before it is assigned to run. Therefore, the effort of speeding up processing time would be wasted if the time that the job remains in the queue is incremented [16]. Since the time in the queue depends on the queuing techniques applied, some turnaround measures are only considered from the moment the job is allocated. The most common way to measure the turnaround time of a system is to calculate the mean turnaround time (MTAT) [81]. MTAT is defined as,

$$MTAT = \frac{\sum_{i=1}^{N}(t_e - t_s)_i}{N}, \tag{4.9}$$

where t_e is the time when the job was executed, t_s is the time when the job started, i represents the present job that is been measured, and N represents the total of jobs measured.

If turnaround times are a major impact on business, then any increment on system performance will generate more gain in productivity. For example, a reduction of the turnaround times of the design cycle of a product will improve the date for its market presentation, which can benefit the business by leading the market for that particular product and can have an advantage over competitors who have not brought out a similar product to the market yet.

Scaling a system to improve turnaround times tends to follow certain strategies. According to Shi [71], some researchers in the parallel community have been using Amdahl's Law to obtain estimated speedups as measures of parallel potential. Amdahl's Law estimates that for a fixed problem size the increment on the number of processors will yield a speedup no greater then the time that it takes to solve the serial fraction of the program [3]. From this statement, we can observe in the graph in Figure 4.6 how the unit gain rate of the system is following Amdahl's Law as the performance of the system is increased. Therefore, the unit gain rate $\varphi(P(S))$ will have an upper limit based on the performance upper limit. Meaning that no matter how much investment is done in acquiring more processors, we will not expect more profit beyond this limit. This was expressed earlier in this chapter.

4.1.1.1 Case example: Simulation and Modeling Systems

The following examples represent cases where turnaround times affect the business goals. The cases are based on two well-known industrial fields: automobile and manufacturing.

Car design simulations with computational fluid dynamics (CFD) and computational solid/structural mechanics (CSM).

Organizations in the car industry are dynamic enterprises that comprise many goals. A very important one is reducing the development cycle of their vehicles. Real-life tests are necessary to assure the quality of the product. These tests are very expensive and time-consuming, which is why many car organizations rely on computational fluid dynamics (CFD) and computational solid/structural mechanics (CSM) to perform simulations of these tests [28].

Mercedes-Benz is a well-recognized organization in the car industry. This organization relies heavily on CFD for the simulations of many of its processes, which are highly computation intensive. With the porting of CFD programs to parallel computation they could reduce the turnaround times in two of their main simulations: engine cooling and full car design [28].

Initially, engine cooling models provided 10 simulations in a time frame of six months, where pre- and post-processing were included. Now, with the increment in power of processing of their system (IBM SP2 with 16 processors), they have managed to reduce their turnaround times and increase to 30 through 40 simulations in the same time frame. This increment allowed Mercedes-Benz to gain time in maturing the design process thereby reducing the time by almost 50% for testing and experiments [28].

A similar situation was achieved on full car simulation. The simulation required a mesh of the order of 10 million cells, with a requirement of approximately 6,000 Megabytes.

Initially they used an IBM SP2 system with 8 processors for a simulation that took 21 days. An upgrade of the system to 128 processors made it possible to reduce the turnaround time of the simulation to a timescale between one and two days. This upgrade incremented their productivity and the accuracy in the design of their vehicles. Mercedes-Benz has benefited from the performance of high-accuracy simulations on a day-to-day basis by incrementing the turnaround times of parallel high-performance systems [28].

Crash simulations are carried out by the car industry through CSM. Thompson Ramo Wooldridge, commonly known as TRW, is a very well-recognized supplier of the car industry that develops car safety products among their supplies. Computer simulations are performed at TRW to develop air bag systems with the best suited shape and volume, and the gas generator that will deploy the bag optimally for a given car interior. The simulations include verification runs for complex models and many optimization processes. TRW used to perform these runs in 2 to 3 days. In order to stay competitive and to reduce the production time for their side impact airbag systems, TRW moved to a distributed cluster system and reduced the run times to 20 hours, and as part of the optimization processes, individual simulations can now be performed in 3 (rather than 12) hours [26]. TRW understands that in order to stay in the market as a competitive company and to prepare for anticipated enhancements in modeling, investments on their parallel system are expected in the future.

Polymer Extrusion.

Polymer extrusion is the process of converting raw polymer, usually in the form of powder, to bulk material in the form of small cylinders, which is the way it is sold to customers. The conversion of this material is done by a double-screw extruder machine [27].

Shell introduced in the market a new high performance polymer called *Carilon*. This material had different properties that made it impossible for standard extruder equipment to be used. The testing for optimizing the equipment is very costly since it considers a down-time of the extruder, and a single day can cause losses of up to $100K [27]. Therefore, the use of numerical CFD simulations (based on the POLYFLOW software) was developed and the requirement of high computing as well as memory were limiting factors in the actual system (IBM SP2 node). The simulation took more than one week, which is not an optimal time for a Shell product schedule. By scaling their system using 22 nodes of their SP2 they could reduce the turnaround times to 15 hours. Shell benefits substantially from these results, having accurate results and more productivity into their design cycle [27].

4.1.2 Performance Based on Throughput (Transaction Processing)

The majority of organizations whose system performance depend on throughput are concerned with generating more responses or satisfying as many requests in the shortest time possible. Throughput is a way to identify quantity, how much the system can solve in a

period of time. Thus, many throughput measurements are used in transaction processing systems (DBMS, Web servers, etc.).

Following the end-to-end response time perspective, the measurement of performance for any transaction processing systems should be based on questions like “what a transaction is?”,” how they are counted?”, and “how the business is impacted, if there exist more (less) transactions than planned?” By answering these questions it will be easier to establish the revenue per transaction, and thus, the output of our production function for this domain.

Many authors have defined transaction processing [50, 36], but we find it useful to present it in a practical definition as “a type of computing processing in which the computer responds immediately to the user, sometimes called on-line transaction processing (OLTP)” [84]. Each request is considered a transaction. Transactions are related to database systems and defined as “a unit of program execution that accesses and possibly updates various data items” [73]. Industries that manage, sell, or distribute a product are related to transaction processing. According to Gray and Reuter [36], transactions are in:

- Finance: Automated Teller Machine (ATM) and any other type of point-of-sale terminals.
- Communication: Call setup transaction to generate the billing.
- Manufacturing: Inventory planning, accounting, order processing, etc.
- Others: Travel agencies, airline reservations, etc.

Although the concept of transaction varies from business to business the properties of a transaction must prevail. As described by Allmaraju [2], transactions have the following

properties: atomicity, concurrency, isolation and durability. These, are called the ACID properties and guarantee that a transaction is never incomplete, the data is never inconsistent, concurrent transactions are independent, and that the effects of a transaction are persistent.

Transaction processing businesses (finance, banks, e-commerce, etc.) consider transactions as the core of business, which means that transactions should be managed accurately and rapidly. The idea of handling a high volume of request in a short time frame is the main motivation. The question is: how to measure the performance of transaction processing systems? The most common metrics are presented in transactions per second (tps) or transactions per minute (tpm). A simple way to measure transactions is represented in Equation 4.10 [81]. The higher the TPS result, the greater the $\varphi(TPS)$ and the less the response time of the system (MTAT) for organizations that depend on transaction processing.

$$TPS = \frac{Numberof Transactions}{TotalTimeTaken} \tag{4.10}$$

The use of benchmarks is also a way to evaluate the performance of transaction processing systems. The Transaction Processing Performance Council (TPC) [80] is an organization dedicated to elaborate benchmarks for database and transaction processing applications. The TPC provides information with respect to transaction performance (tpm) and price/performance ($\$/tpm$). For this reason, the TPC benchmarks are commonly used as a reference for evaluating transaction processing systems. The TPC categorizes its bench-

marks according to the application. For example, TPC-C simulates a complete computing scenario where a user executes transactions against a database. The benchmark is based on an order-entry environment. The transactions performed include entering and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. TPC manages other categories (TPC-A, TPC-B, TPC-D, and TPC-W) besides TPC-C[4]. TPC results are good alternatives as a support for comparisons of systems in performance and cost, but like any benchmark their results are based on a particular scenario.

The performance of a transaction processing system depends on more than just CPU speed. Metrics based on Flops or MIPS measure the raw performance of the system based on the CPU not the transaction processing performance. Transaction processing systems depend on many factors besides CPU speed (application, memory, I/O devices and bandwidth), and they should be taken into consideration when measuring the performance in this domain.

4.1.2.1 Case Example: Transaction Processing Systems

As mentioned before there are many cases where transaction processing systems are applied. Below we discuss in more detail a model for a transaction processing system that has revolutionized the way to do business.

---

[4]http://www.tpc.org

E-Business.

Thanks to the evolution of the Internet, many businesses have changed their strategy toward customers by serving and providing them electronically. This strategy is known as e-business. In its simplest form, e-business can be developed by the creation of a website that displays the services and information that the business offers to its customers. The more business applications are involved via the web, the more productive the business will be in terms of its Internet strategy, but this is not always an easy task [68].

Doing e-business requires that all transactions are performed via the web, generating electronic transaction processing called e-transactions. The measuring of e-transactions depends on many factors: bandwidth, ISP, web server, database server, etc. The main interest is on end-to-end response time of the e-transaction, which can be achieved by applying analysis like the Response Time Pipe (RTP) [61]. RTP represents the response time of an e-transaction by several measurement paths, and each path will represent a portion of components that are involved in the overall transaction. For example, Figure 4.7 represents the implementation of a simulated RTP analysis. Each of the points represent a measurement of a component included in the total response time of the e-transaction (Equation (4.11)). $WNET$, represents the response time of the wide network (between the user and the host web server), $HWEB$ represents the response time of the host web server, $LNET$ represents the response time of the local network (between the host web server an the database server), and HDB represents the respond time of the database server

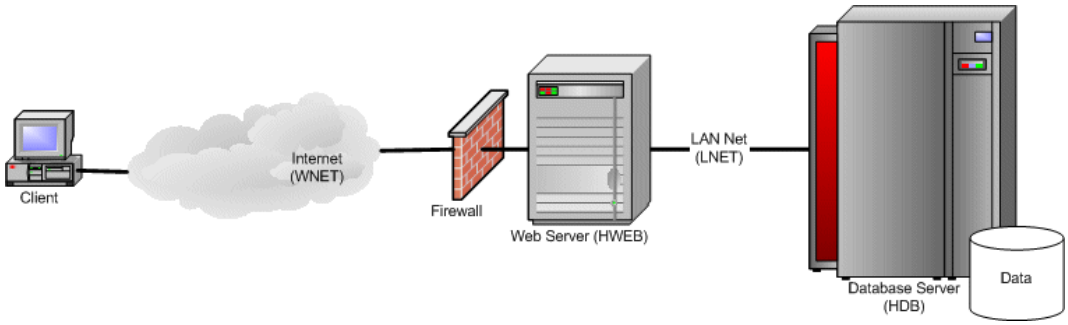

Figure 4.7 Representation of a response time pipe in a three-tier architecture

$$RTP = WNET + HWEB + LNET + HDB \tag{4.11}$$

RTP can contain as many points of measurement as necessary, as long as these points of measure have an impact on the total response time of the e-transaction. For example, a corporate firewall, which resides between the end user and the web server, might have very little intervention in the overall process of the e-transaction, and any improvements on this process will not generate additional benefits to the throughput of the business.

There are many e-business models ranging from simple to advanced [68]. The simplest e-business model involves business transactions that are restricted to filling in and mailing an on-line form. For this type of system no more than 200 simple transactions per day are handled and business value is low. The RTP analysis for this scenario will have very few points of measure.

In an intermediate e-business system, transactions become interactive. Volumes increase because of the interoperability between the end user and the site; the number of

events that happen in a single intermediate transaction range from five to ten as a user enters the site and puts something in the shopping cart, and the system run checks against the inventory and calculates shipping costs. Such systems can easily be required to handle 200 transactions per hour, and business returns are in the 5-25% range. Each transaction involves a larger amount of information. However, orders are still typically batched and then moved from one system to another for overnight processing.

Advanced e-business opens the possibility of negotiating and collaborative opportunities in real business-to-business (B2B) transactions. In such a system, requirements can be as high as handling 400 e-transactions per second over the Web. Business value is enormous for these systems since they permit reducing inventory, cost of order entry, scheduling costs and permit moving to a real time delivery model.

Moving from simple e-business to advanced e-business creates additional value for customers and business, thus increasing transaction volumes. As the value to business and customers increases, the qualities of service, such as response time, demanded to the system also increases.

4.1.3 Performance Based on Scale-up (Larger Problems)

For research, academic and government areas, the main reason for parallel computing is to solve larger problems, problems that cannot be solved with regular serial computing and that require great amount of computer resources to generate accurate results. For example, in many simulation models (automotive, airplane, etc.) the solutions to linear equations

have a big impact on the overall run time; the results of a 10 by 10 mesh might not have the same high resolution results of a 1000 by 1000 mesh. In a case where the simulation of the the former takes one hour compared with one day for the latter, the application scientist might be forced to narrow the scope of his analysis because each run takes too long, losing important and accurate information. Therefore, it is important to evaluate parallel systems in terms of the scalability they have toward an increment on problem size.

Previously, we mentioned Amdahl's Law as a way to predict the performance of a parallel system. According to this law if the number of processors is increased while the problem size is fixed, performance is limited to the a serial fraction of the application. There exists another approach to performance scalability of parallel systems that debates Amdahl's argument. This argument was made by John L. Gustafson [39] and states that by increasing the problem size and the number of processors the performance can be increased by more thana fraction of serial processing. The reason behind this argument is that by adding more resources and increasing the workload, the workload will likely expand to make use of the facilities, generating more parallel processing.

Scale-up systems, can have a dramatic impact on profit if the business is centered in solving larger problems, however the production function for this cases is usually difficult to obtain due to the nature of the problem. For example the performance model may be presented as size units that determine the size of the problems solved by the system and the unit gain rate will be expressed by the value according to the size. The size and time constraint are not easy to determine in this cases. For this reason, we find difficult to

quantify the performance gain for this domain. In these cases the business will benefit from intangible factors, such as:

- Recognition: The solution to large problems can have high recognition in scientific, academic and government circles (*e.g.* Gordon Bell Prize, Top 500 list).
- Contribution: Well-recognized organizations in HPC (IBM, NSF, DOD, etc.) are willing to give grants in return for solutions to high degree problems.
- Demand: The increment in resources to solve bigger problems will also have an increment on different types of users that will also benefit from the solution of similar problem size in this platform.
- Quality: In the design cycle of a product, simulations are often used, and with the solution of bigger problems accurate results can generate better product quality for the market.

4.1.3.1 Case Example: Molecular Simulation

Molecular simulations are primarily done by molecular dynamic (MD) modeling. Many MD applications have been designed to simulate fluids, solids and macromolecules at an atomistic level of detail. MD simulations require intensive and complex computation. For this reason, parallel systems are used for their implementation. There are several parallel algorithm for MD simulations [8]. The Parallel Computer Science group at Sandia National Laboratories [69] performed an experiment based on MD applications to verify and compare the ability to scale on two different parallel architectures: massive parallel processing system (MPP Intel TFlops) and Linux Cluster (DEC CPlant). The experiment was based on a scale-size problem, starting at 32,000 atoms per processor, doubling the problem size as the number of processors are doubled. Table 4.2 shows the values of speed and efficiency obtained from scaling the system [69].

Table 4.2 MPP vs. Cluster

Processors (P)	1	2	4	8	16	32	64	128	256	512	1024
Problem Size (N)	32K	64K	128K	256K	512K	1.02M	2.05M	4.10M	8.19M	16.4M	32.7M
Tflops CPU Speed	0.534	0.538	0.549	0.557	0.558	0.564	0.563	0.566	0.577	0.574	0.591
Tflops Eff. (%)	100.0	99.3	97.3	95.9	95.7	94.7	94.8	94.3	92.5	93.0	90.4
CPlant CPU Speed	0.225	0.237	0.249	0.260	0.270	0.288	0.303	0.300	0.315	0.310	—
CPlant Eff. (%)	100.0	94.9	90.4	86.5	83.3	78.1	74.3	75.0	71.4	72.6	—

We can see from this experiment how MD applications have good scalability on both architectures. We can also see how efficiency is maintained between 90 and 100% for the MPP Intel TFlops system, which is highly scalable, although the CPlant system has better performance. This experiment was also performed for a fixed-size problem (32K) where the number of processors were increased from 1 to 1024, the results showed a less scalable systems compared to the scaled-size problem.

4.2 Other $\varphi(P(S))$ Representations

In the previous section we discussed how to express the unit gain rate φ from the system performance $P(S)$. As mentioned, $\varphi(P(S))$ is a representation of the profit obtained as the result of the work performed by the system in one unit of time. In organizations that are service-driven and process-centric (E-business, Telecommunication, etc.), where technology constitutes the core without which the business could not exist, the representation of profit is quantifiable. However, for many organizations, profit is difficult to quantify. Areas like research, education and government applications are often provided by non-profit organizations. Moreover, in many organizations the technology department has a support

role rather than a service role, which means that there is little or no direct impact of the technology over the business. In such organizations, profit is also difficult to quantify. This section will provide a view of those intangible benefits obtained from system performance, that also have an impact (not directly) on business profit and should be evaluated when making decisions for replacing or upgrading the technology used.

Determining the amount of profit obtained from system performance is to express its return on investment (ROI). ROI is the measure on which many organizations rely to understand how well the business is doing. ROI on IT is a new concept, but from the evolution that IT has had in many industries, calculating ROI is not an option anymore. In a survey done by Network Magazine [53], almost 35% of IT managers consider ROI as an important decision-making parameter for IT investment. Unfortunately, ROI on IT is not always in the form of hard cash. ROI in many cases may be intangible, and therefore invisible. In spite of such constraints of ROI on IT there are other ways that returns on IT can impact the business [53]:

Improvement on efficiency: Efficiency can be improved by investing in a more reliable and accurate system that can improve the business processes, reduce communication, offer a good response time, and increase productivity. For example, the use of email servers decreased the amount of the paper being used and reduced the cost of phone bills.

Quality of work: The use of better and more sophisticated tools increment the quality of work. There exist many sophisticated applications that bring highly accurate informa-

tion that can greatly benefit the business, but if they do not have the qualified architecture to support them, they are useless[5].

Customer satisfaction: For many IT departments the real customers are inside users from the same organization. They require that their processes run effectively so they can perform their job. Their satisfaction is a measure for IT performance. Surveys are one way to measure customer satisfaction. An increment on projects and the capture of new users is another way to represent an improvement on IT performance.

HR reduction: Reduction of staff is a way (less likely) to reduce cost. By automating processes and standardizing applications, it is possible to reduce the number of staff. A not-so-complex system requires a small team to administer it. As a consequence, a better training program can be devised, *i.e.*, training will be more effective because of the reduced group, lowering training cost.

Indirect Returns: Indirect returns are returns that can be reflected in other areas besides IT but bring profit to the organization. For example, reduction on production cycles, lesser time to market a product, cost savings, etc. The IT department interacts with many business processes that depend on system performance. It is important, before making any decisions over the system, to know and understand the interaction between the system and the multiple processes that will benefit from it. What benefits one process might not benefit the others. Taking these facts into consideration will enhance productivity and overall the business goals.

[5]Implementations of Enterprise Resource Planning (ERP) or Customer Relationship Management (CRM).

It is worth noting that these benefits are usually accounted over a period of time, three years being the industry norm [53]. Calculating ROI gives relevant information to the business.It can be used for dual purposes: to develop a business case for justification of investment in a particular vertical or project and also to gauge how well an enterprise is managed.

CHAPTER V

THE COST FUNCTION

In chapter III, we described the cost/profit effective metric as a metric constituted by two functions, the $Pro(P(S_p), t)$ and the $Cost(S_p, t)$ function. The former was described in the previous chapter. The latter will be the main topic of this chapter. In order to develop the cost function, we first need to identify the factors that are involved in the cost of parallel computing. The true cost of computing (parallel or serial), has had an increasing interest among IT analysts during the last few years. With the evolution of technology, more and more organizations worry about their technology investments and where the money is spent. According to some analysts the calculation of cost in computing has been misunderstood over the years [77]. As a consequence, there is an inability to convincingly demonstrate that investments in technology have resulted in measurable productivity improvements. This fact is called the productivity paradox phenomenon [77]. Today, with the use of economic models (Total Cost of Ownership (TCO), Total Economic Impact (TEI), Cost/Benefit Analysis (CBA), etc.), it is possible to analyze the cost in computing. Moreover, these models make analyses of cost savings and present the risks and benefits of investing on a certain technology.

Calculating the true cost of computing (parallel or serial) is a complex business because circumstances vary. Purchase costs are fairly clear and often users will consider nothing else in determining the economics of their system. Running costs are less easy to determine and often ignored, yet they are quite significant for all but the simplest of systems. When these running costs are taken into account they are frequently underestimated. Choosing a vendor on the basis of purchase price alone is generally a short-sighted decision. Where increased system complexity exists, such as with multi-user and networked systems, running costs become even more significant [48].

In this chapter we will define the cost function, $Cost(S_p, t)$, based on the factors that are displayed in Figure 5.1. From previous analyses [48, 19, 62], we classified the cost of parallel computation into two main costs: explicit and opportunity costs. Explicit costs are easy to quantify, and they are mainly identified in the purchase process of the system. Opportunity costs are often hidden and generated during the useful life of the system. Opportunity costs are the result of time wasted and productivity lost. In the following sections we will examine in detail these two classifications according to Figure 5.1, and then we will give a formal (mathematical) definition of the $Cost(S_p, t)$ function as part of the cost/profit effective metric.

5.1 Explicit Costs

The explicit costs, as mentioned previously, correspond to costs that are mainly identified in the purchase process: hardware (computing nodes, processors, memory, cables, disk,

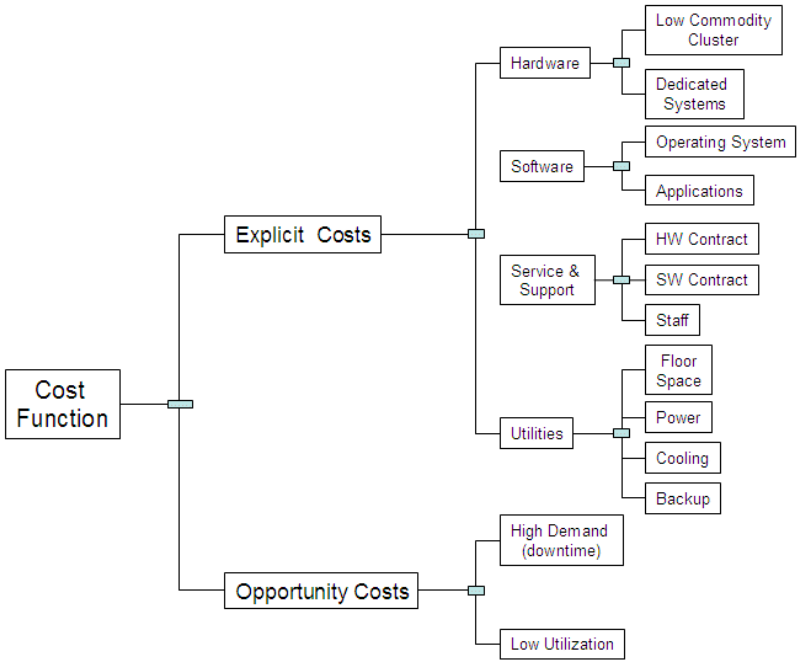

Figure 5.1 Factors that determine the Cost Function

etc.), software (applications, operating system, etc), service and support contracts, and utilities (equipment accommodation and power), as shown in Figure 5.1.

5.1.1 Hardware Cost

Hardware costs are accounted at the moment the system is acquired, by lease or by total purchase of the system. The industry generally considers original purchase costs to be 10 to 35 % of the total cost of ownership of a computer system. Hardware costs of parallel computing depends on the system architecture. We classified the cost of hardware, for parallel computing, into two main categories according to the system architecture: cluster systems, based on commodity processors (computation distributed in many boxes) and

dedicated systems (traditional parallel computing, delivered in one box, often called “big iron”). The hybrid case (cluster systems with dedicated servers) is not considered for cost analysis since we can derive it by merging the studies from these two classifications.

5.1.1.1 Hardware costs of commodity cluster systems

The total cost of a commodity cluster system is divided into the cost of the components that constitute the cluster. The components of any cluster system are:

Nodes: A node in a cluster is defined as an element with a CPU, memory and disk. Nodes are divided into two types, the head nodes and the computation nodes. The head nodes are used for authentication, allocation and resource-sharing processes. They are usually viewed as a console with a monitor and keyboard components. The computation nodes are used strictly for computation, and are not attached to monitor or keyboard. Any access to the computation nodes is done through the head nodes. The head nodes can have less computation power than the computation nodes (less processors, processor speed and/or memory).

Network: A communication link must be provided to permit the nodes to interact together as a whole. The network in a cluster is a crucial component. If the network does not provide enough speed for communication, the latency will be too unbearable to run any computation. There are many ways to design the network for a cluster, depending on what is needed. For example, if the workload of the system is based on 80% computation, then an average network composed of a Fast Ethernet (100Mbps/sec) connection will probably

do. However, if the workload of a system is based on 80% communication, then a high speed network, like the use of a gigabit switch, will be required. Depending on the need, the cost of network components will vary. A balance between the workload's necessities for communication and computation is recommended to have an adequate cluster that can support any type of load.

Miscellaneous: Components that correspond to cables, racks, and additional cards correspond to the miscellaneous components of a cluster system. Although they are not costly with respect to other components of the cluster, they are part of it and should be considered when calculating the total costs. On average, they represent between 10 and 15% of the purchase cost per node of a cluster [19].

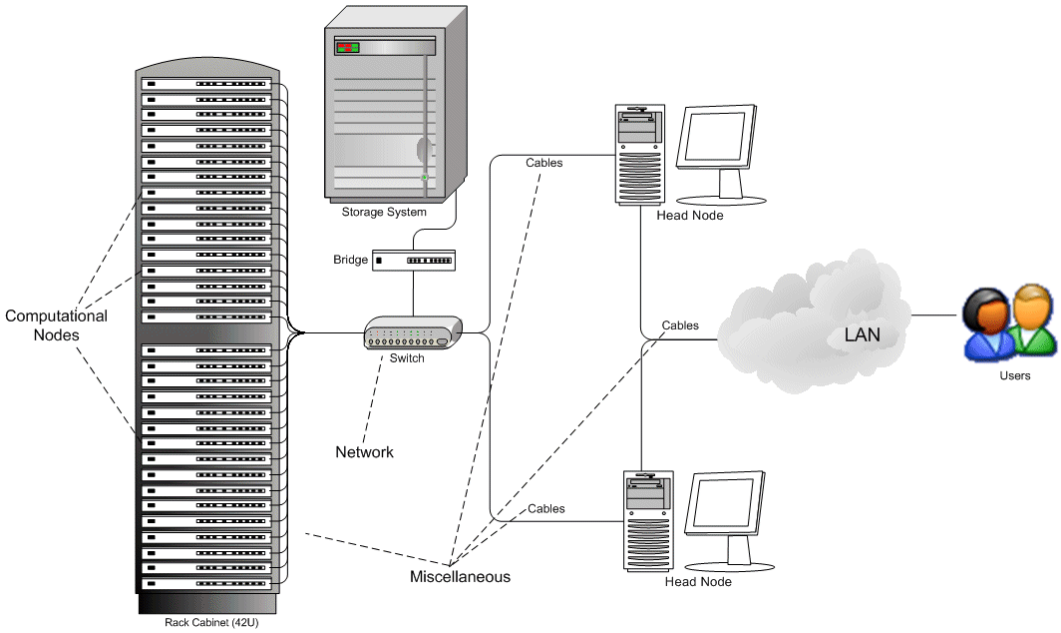

Figure 5.2 Graphical representation of a cluster system

Figure 5.2 shows a graphical representation of a cluster system. On average the purchase cost of a cluster is $4,000 per Gflop [19]. This price only comprises the cost of computation nodes; the total cost of a cluster per Gflop should include the network and miscellaneous costs. Since these costs vary according to the size of the cluster, they are not included in many market analyses.

5.1.1.2 Hardware costs in dedicated systems

Dedicated systems correspond to the traditional high performance computing systems, which are sometimes called “big iron”. They are delivered through proprietary architecture and operating systems and many were the pioneers of high performance computation (Cray Systems, IBM S/390, etc.). Dedicated systems are based on Vector, SMP or NUMA architectures, although some distributed systems are built in one box. For example, the IBM SP2 with SMP systems interconnected with a high switch interconnection (hybrid system). Independent of the architecture, generally, traditional parallel systems are composed of:

- Processors: High class processors (regularly based on RISC architecture), with a speed and price greater than commodity processors. Usually dedicated system processors are provided in powers of two (*e.g.* 8, 16, 32, 64, etc.).
- Memory: Memory is shared through processors in dedicated systems (SMP and NUMA architectures). They range from 1Gb up to 128Gb, providing a large pool of memory to be accessed.
- Interconnection: A high-speed interconnection resides inside the architecture of dedicated systems. There are different types of interconnections according to the architecture (crossbar switch, bus, mesh, cube, etc.).

- I/O: A broad bandwidth for I/O is also provided in these types of systems, although there exists storage management systems (EMC, Clariion, etc.) that are independent of dedicated systems, these systems still provide fast access to internal data.

Dedicated systems tend to be more costly than commodity cluster systems, but they also tend to provide more performance with less processors. This tendency is changing as a consequence of Moore's Law. On average the purchase cost of a dedicated system ranges from $10,000 to $25,000 per Gflop [19]. Also because of Moore's Law, the price-to-performance ratio of a cluster and a dedicated system decreases exponentially with time. Some analyses indicate that there is a factor of two in advantage of the price-to-performance of commodity cluster systems with respect to dedicated systems [58]. This behavior is illustrated in Figure 5.3, where we can also observe that although commodity clusters and dedicated systems depreciate almost at the same rate, the upgrade cycles (time to renew the system's processor) of dedicated systems can take as much as twice the time that it takes for commodity cluster systems to upgrade. The gap exists because commodity nodes are more affordable than processors on dedicated systems.

5.1.2 Software Cost

The software cost is dependent of the hardware architecture where it is going to be implemented. For example, a dedicated system with an SMP architecture will find several commercial applications in the market, but for cluster systems many of these applications must be ported and probably are not yet ready to be implemented in cluster systems because they do not have a shared memory architecture. Software costs include: the op-

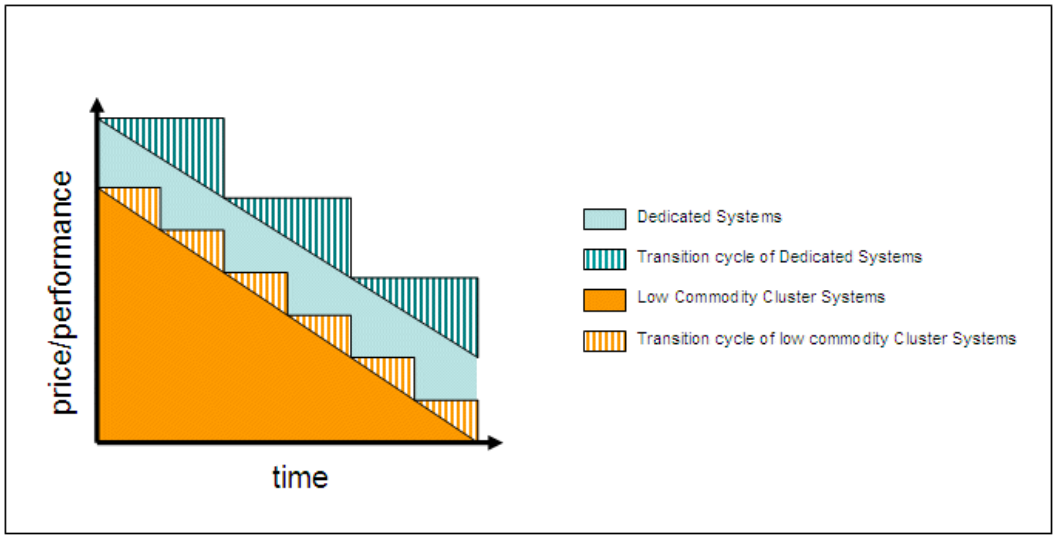

Figure 5.3 Price-to-Performance graph of dedicated and cluster systems

erating system (Linux, Windows 2000, Solaris, AIX, Unix, etc.), queuing software (NQS, MDVS, PBS, etc.), applications, job monitor, etc. Some commercial applications have licensing costs, some per node (in the case of cluster systems). On average, the cost of software ranges between $500 to $1,500 per processor [19].

5.1.3 Service and Support Cost

Services and support costs are divided into the maintenance contract of hardware and software, and staff support.

- Hardware maintenance: Involves installation, specialized support and training of the system. Sometimes this cost is included in the purchase price of the system (for example, Dell includes a 3-year basic maintenance contract in the purchase of their servers). This cost will vary according to the vendor. Some will offer separate maintenance contracts according to service levels, for example gold (a year of on-site support, inventory maintenance for spare parts, and web line support), silver (same

day support on-site), bronze (next business day support) and so on[1]. Usually hardware maintenance contracts range from 20 to 25% of the total cost of purchase in cluster systems. For dedicated systems the hardware maintenance contract exceeds 35% annually.

- Software maintenance: Depends on the application and vendor, and involves upgrades to new versions, licensing and staff training. On average, software contracts typically cost 20% of the software purchase annually.
- Staff: Represents the annual salary of the IT professional that administers the system. For example, a cluster system with 300 nodes and two IT professionals managing the system full time, with a salary of approximately $65K annually, has an annual cost of labor of approximately $430 per node.

5.1.4 Utilities

Utilities refers to all the costs incurred for the system accommodation in the IT environment. This includes the following:

- Floor space: The floor space is designed for fire control, environmental control, security control and to support the weight of the systems. A proper configuration of floor space ranges between $17 and $25 per square foot annually [19]. As an example, let us consider the cost of floor space at $21 per square foot annually. For a cluster system configured in a rack cabinet with 32 processors, the annual cost is $140 per 7 square feet of floor space usage[2]. A similar configuration on a dedicated system (24 processors) costs $186 per 9 square feet of floor space usage[3].
- Power: The power costs of the system may vary according to the configuration. On a single processor, power consumption is between 20 and 60 watts, resulting in a cost between $20 and $70 of annual usage [19, 65].
- Cooling: The systems generate a great amount of heat that must be lowered by cooling systems like air conditioners. As an example, a cluster of one thousand processors requires two 24-tons of air conditioning, at $123 per ton, resulting in $6,000 of annual system cooling [19].

---

[1]This is the way Compaq categorizes their service levels. Other vendors may vary in nomenclature and periods.

[2]Prices based on Dell's cabinet configuration.

[3]Prices based on Sun Microsystem's server configuration.

- Backup: The backup systems are the power generators that are in use in case of failure of the energy supply. They can cost from hundred to thousands of dollars depending on the power consumption needs of the system.

5.2 Opportunity Costs

In the previous section we discussed the costs that are involved with all the components that constitute the purchase and operation of a parallel system. In this section we will discuss the costs that are present when the parallel system is not in operation or is under-utilized. These costs are called opportunity costs and are present in the running time of the system. Therefore, they are not viewed until the system is in operation. We created two exclusive categories to identify the opportunity costs:

Cost for low utilization of the system: The cost for low utilization of the resources of the system occurs when the system is not fully utilized, in average, less than 70% to 80% of utilization. Thus, under utilization causes processors to be idle at some point and as a consequence, the efficiency of the system will decrease and the costs will increment. For example, say a company needing 100 Gflops of processing power purchased a cluster or a dedicated system for $400,000. Suppose also that the peak computational power of the system is 100 Gflops. At this power level, the computer is capable of performing 8,640 trillion calculations per day. If the company operates the computer constantly, the cost of this computation would be 7 cents per trillion calculations[4] [12]. However, if the company uses the machine only three days each month, the cost of computation would increase to

[4]Based on a three year amortization at 12% interests, plus 12% of hardware costs for operational expenses.

70 cents per trillion calculations. Even for a large firm that can afford a supercomputer, having only intermittent needs that will leave the machine idle for extended periods of time, presents an inefficient use of resources and an increment in cost. We formulate the cost of under utilization of a system as,

$$\begin{aligned} \rho_u &= (1 - f_u) \times M \\ f_u &= \tfrac{W_r}{W_{op}} \\ W_r &= t_r \times P \\ W_{op} &= t_{op} \times P. \end{aligned} \tag{5.1}$$

In Equation 5.1, ρ_u represents the cost of the system under utilization, M is the cost of the system during its life cycle. W_{op} represents the optimal number of operations performed when the system is fully utilized (t_{op}). W_r represents the actual number of operations that the system performs in the real time the system has been utilize (t_r), and P the performance of the system measured in flops.

As the real time utilization, t_r, of the system increments, then the cost per under utilization of system resources will decreases as shown in Equation 5.1. The cost ρ_u is calculated once the system is in production, and it will tend to decrease as soon as the utilization of the system stabilizes (users adaptation to the new system, applications maturity, etc.).

Cost of system downtime:

A system failure can be represented in two ways, when it produces incorrect results or does not produce a result at all. Either way, these are losses and they represent money. Each organization has its own way to determine the financial impact of system downtime

(loss of sales, employee's productivity, potential customers, drop in manufacturing product, etc.). There are two main causes system downtime:

- Predicted: This time is scheduled by administrators for maintenance. It includes system upgrade and backup. Generally, IT administrators schedule the time for maintenance during times that interfere as little as possible with production tasks (over night, during weekends, etc.).
- Unpredicted: This time is the result of system failure. Typical system failures are caused by hardware, applications, power supply and human intervention.

In a report by Techwise Research [79], a survey was performed among 93 IT professionals across the top industries (telecommunication, manufacturing, finance, health care, government and service) to analyze TCO of enterprise cluster systems, based on RISC architecture, according to their downtime cost-per-hour. Enterprise systems are, according to the International Data Corporation (IDC) [45] categorization [12], those systems that cost between $300K to $1M. The respondents rated reliability as the most important factors for purchase decision of cluster systems, closely followed by performance and vendor reputation. Purchase price was rated as one of the last factors. As we can see, the downtime of a system is a crucial factor that have a financial impact on the business. From this survey, on average, respondents said that each hour of downtime in their organization cost a total of $71,000. By reducing the downtime cost of a system its TCO will reduce in great proportion. For example, we found as part of the result of this survey that the best system had an average annual downtime of 6 hours, resulting in more than $4M annually in cost.

5.3 Formal definition of the $Cost(S_p, t)$ function

In this section we will give a formal definition of the cost function, $Cost(S_p, t)$, using the concepts discussed in earlier sections of this chapter and represented in Figure 5.1. As mentioned earlier we categorized the cost function in two main functions, explicit and opportunity costs. We will give the formal definition of each of these functions before giving the formal definition of the cost function. As we did in chapter IV, we will replace the notation S_p with S to simplify notation.

5.3.1 Explicit Cost $\Sigma_c(S, t)$

We define the explicit cost function as,

$$\Sigma_c(S, t) = C_H(S, t) + C_S(S, t) + C_{Su}(S, t) + C_U(S, t). \tag{5.2}$$

In the Equation 5.2, C_H represents the hardware cost as described in section 5.1.1, C_S is the software cost as described in section 5.1.2, C_{Su} is the service/support cost as described in section 5.1.3, and C_U is the utilities cost as described in section 5.1.4. Each of these terms are defined for a configuration S over time t. We will now define each of these terms.

The software cost C_S depends on the licensing agreements of the software vendor, usually managed as basic licensing plus additional user/node licenses. The service/support cost C_{Su} considers factors such as training costs, staff salary (annually), and hardware and software contracts (annually). The utility cost C_U considers factors such as cost per square

foot, cost/power, etc. For more details about the components that constitute these costs, please refer to their corresponding sections as indicated in the previous paragraph. The hardware cost C_H is defined by the equation:

$$C_H(S,t) = C_A(S,t) + f_d(S,t)$$

$$C_A(S,t) = \begin{cases} \sum_{i=1}^{hn} C_{HN_i}(t) + \sum_{i=1}^{cn} C_{CN_i}(t) + C_\eta(t) + \mu_c(t) & \text{if } S \text{ is a Cluster system} \\ C_{proc}(t) \times p + C_{mem}(t) + \mu_d(t) & \text{if } S \text{ is a Dedicated system} \end{cases}$$

$$f_d(S,t) = \begin{cases} \frac{C_A(S,t)}{L_f(t)} \times \delta & \text{if } S \text{ is a Cluster system} \\ \frac{C_A(S,t)}{L_f(t)} & \text{if } S \text{ is a Dedicated system.} \end{cases} \tag{5.3}$$

The architecture cost C_A is the cost of the hardware for system S (cluster or dedicated system). For a cluster system, the cost is composed of the cost of the head nodes (C_{HN}), the cost of computation nodes (C_{CN}), the cost of the network (C_η), and the cost of the miscellaneous components (μ_c). For a dedicated system, the cost is composed of the cost of the processors (C_{proc}), the cost of the memory (C_{mem}), and the cost of miscellaneous components for the dedicated system (μ_d). The other factor of hardware cost is the depreciation cost f_d, which is defined as a ratio of the architecture cost C_A over the lifetime of the system L_f. Since only specific components of a cluster system depreciate over time (*e.g.*, network, cables, etc. do not depreciate at the same rate as the computation compo-

nents), and in order to avoid the use of more economic terminology, we will assume that cluster systems only depreciate by a factor δ of the total architecture cost. Depreciation is calculated as *straight* depreciation calculation, as described in chapter III, section 3.4.

5.3.2 *Opportunity Cost* $\Theta_c(S,t)$

The opportunity cost is defined as,

$$\Theta_c(S,t) = \rho_u(S,t) + C_d(S,t), \tag{5.4}$$

where ρ_u is the cost of under utilization as defined in Equation 5.1 and C_d is the cost of system downtime, which is defined as,

$$C_d(S,t) = \varphi(P(S)) \times (t_{pred} + t_{unpred}), \tag{5.5}$$

where $\varphi(P(S))$ is defined in section 4.1 as the unit gain rate (in monetary units) of the performance P of a system with configuration S. t_{pred} and t_{unpred} are the predicted and unpredicted system downtime, respectively, as described in section 5.2.

5.3.3 *Formal definition of* $Cost(S,t)$

Given the definitions of the explicit and opportunity cost functions we can now formalized the cost function as,

$$Cost(S,t) = \Sigma_c(S,t) + \Theta_c(S,t). \tag{5.6}$$

In Equation 5.6, Σ and Θ represent the explicit and opportunity costs, respectively. In chapter VI, we will apply the $Cost(S,t)$ function to a study case, as formulated in this chapter.

CHAPTER VI

CASE OF STUDY

In this chapter, we present two cases of study for the application of the cost/profit effective metric and analyze the results to inform the decision of upgrading or replacing the parallel system in each case.

Our cases are based on two real life scenarios: the Cornell Theory Center (CTC) at Cornell University and the Engineering Research Center (ERC) at Mississippi State University. Both centers provide high performance computation and research for a wide variety of community users and corporate clients that require leading edge computational resources. In the past, both centers have made dramatic system changes to their main parallel environment. In this chapter, we analyze these previous experiences as an example to apply the cost/profit effective metric.

We will start by giving a brief description of these two centers and their high performance computing environment. Then, we will give a detailed description of their past and present system configurations and apply the cost/profit effective metric to them. Finally, we will analyze these results to support their decision of upgrading or replacing the systems in each case.

6.1 Background Information

In this section we provide a brief description of the Cornell Theory Center (CTC) and the Engineering Research Center (ERC). Both centers have had the experience of making a transition from dedicated systems to commodity cluster systems to meet their high performance computation requirements with a cost efficient solution. We find it interesting to model both scenarios using similar configurations and then apply the cost/profit effective metric to analyze these systems and their transitions in terms of cost and profit.

6.1.1 The Cornell Theory Center

The Cornell Theory Center (CTC), is a high-performance computing center located on the Ithaca campus of Cornell University [14]. This center supports scientific and engineering research projects across a variety of disciplines, including biology, behavioral and social sciences, computer science, engineering, geosciences, mathematics, physical sciences, and business. For over ten years the CTC ran dedicated systems based on proprietary Unix architecture. In 1997, the CTC decided to search for a more cost effective solution to provide high performance computation to its users while improving the reliability and performance. In 1999, they made their decision toward commodity Intel clusters with Windows 2000. The cluster serves as a production high-performance computing resource for CTC's research community and the Advanced Cluster Computing Consortium (AC^3). AC^3 is a Cornell research and IT service consortium established for corporate, higher-

education, and government agencies interested in the effective planning, implementation, and performance of commodity-based software, systems, and tools [17].

CTC evaluated carefully all the aspects of performance, reliability and cost of the cluster before coming to a final decision. The transition and risks taken by the CTC to commodity cluster systems have resulted in several benefits that include the growth of community users, management of a standard environment, performance on demand, and increasing research to provide other methods for high performance computation [51].

6.1.2 The Engineering Research Center

The Engineering Research Center (ERC), is located at Mississippi State University [25] and helps support a variety cross-disciplinary research thrusts, such as visualization analysis and imaging laboratory, computational simulation and design center, computational geospatial center, center for computational systems, and the center for DOD programming environment and training. High performance computing needs are served through a variety of systems such as, workstations, dedicated systems and commodity clusters. This last one is the main core for providing high performance computing at the ERC. This cluster system was recently acquired (in year 2000) as a transition from a dedicated system. They have obtained several benefits from this new system, specially on price/performance. They look forward to expanding their platform and maintaining the service of high performance computational excellence to their community users.

6.2 System Configuration

The analysis for these two cases will start by describing the configuration architecture of each system (dedicated and cluster system). We will first describe the system configuration at the CTC case, going from their old dedicated system to the configuration of the cluster system. Then, we will make the same description with the ERC case.

6.2.1 CTC's system configuration

The CTC made a transition in 1999 to replace their old platform based on a dedicated system, IBM SP2 system, to a commodity Intel cluster platform based on Dell servers and Windows 2000. In the next sections we will make a brief description of the relevant components present on each configuration and some of the drawbacks found on the previous system.

6.2.1.1 Previous System: IBM SP2

Like mentioned before, the CTC ran, for over ten years, proprietary Unix-based systems. An SP2 system, from IBM [44], was the core of CTC's technology at the time, using an IBM SP3 switch which provided a high speed interconnection. CTC was seeking new ways to obtain better performance with this system, but they found some obstacles with their configuration [51, 66]:

- Slow cycles for upgrades: The system demand was growing rapidly. To meet these demands the IBM SP2 system needed to be expanded. The upgrade cycles took time (approximately 6 months or more). The spare parts (processors, memory, etc.)

were expensive and the budget had to be planned. Some projects were delayed until resources were available.

- High annual maintenance fee: The maintenance contract for the IBM SP2 system was very high, increasing the system's TCO.
- Head counts: Approximately 12 highly skilled personnel were part of the staff that managed the system at the moment. This also had an impact on the system's TCO.
- Interoperability: An heterogeneous data center is difficult to manage. Systems have to continuously operate in several environments having the user to function between the role of scientist and computer scientist, to develop his/her work.

The configuration of the system under discussion at the CTC is shown in Table 6.1 [25].

Table 6.1 CTC IBM SP2 system configuration

Quantity	**Description**
160	IBM SP2 System Nodes: • 144 Thin Nodes 120Mhz with 256MB Memory RAM • 16 Wide Nodes 135Mhz with 512MB Memory RAM • IBM SP3 Switch Adapter (150MB/sec.)

As we can observe from the table, the IBM SP2 was composed of wide and thin nodes. Wide nodes have approximately four times the cache and memory bandwidth (between the cache and the memory), than thin nodes [44]. Each thin and wide node, according to the configuration in Table 6.1, delivered a peak performance of 480 Mflops and 540 Mflops respectively. An overall of 77 Gflops of peak performance was obtained on this system[1].

[1] Although it was listed in the top 500 list in 1998 at position #76 with 53 Gflops (http://www.top500.org)

Applications were provided to run on the IBM Unix version, AIX. Some of these applications were C, C++, Fortran, etc. For parallel programming, Parallel Virtual Machine (PVM) libraries were available (a version of Message Passing Interface (MPI) libraries). They used IBM Load Leveler as a queue system for the allocation of jobs.

6.2.1.2 The New Solution: Dell Wintel Cluster System

After a careful evaluation of performance, reliability and cost, the CTC decided to decommission their IBM dedicated system for a commodity cluster based on Dell Intel nodes and Windows 2000 operating system. The key factors that made this transition successful were based on [51, 66]:

- Scheduling technology: At the time, the job scheduling systems under Windows were few and expensive. The CTC staff decided to create their own job scheduling system, Cluster CoNtroller, based on Windows 2000. This permitted the system to scale better and work more efficiently.
- Programming Tools: The CTC staff also managed to provide all the programming tools that were available on their existent system (IBM SP2). They managed to find the appropriate math libraries, as well as the versions under Windows for Fortran, C, and C++ compilers.
- User interface: The CTC staff provided an interface similar to the one that users were using in the IBM system. They also provided a Unix-like environment on top of Windows to facilitate the transition to the new environment.
- Migration: The CTC staff provided labs for users to port their code to the new environment. This made possible to handle the needs of users on a case-by-case basis before having the whole system in production.

The configuration of the commodity cluster system, called Velocity I, is shown in Table 6.2 [25]. The hardware installation of this cluster was completed under 24 hours [15]. In approximately eight months, the CTC had the new environment running completely and

in production with more than 90% of the users working on it. The system provided a peak performance of 122 Gflops. Currently this cluster has expanded to approximately 900 processors, and has been partitioned into several clusters according to different projects.

Table 6.2 Velocity I system configuration

Quantity	**Description**
64	Computational Nodes: Dell Poweredge 6350 Server • Quad Processor Intel Pentium III Xeon 500Mhz • 2MB Cache/Processor • 4GB Memory RAM • 54GB Internal Storage Disk
40 128	Network Components: • Emulex Xilan Switch GNX5000 • 8 Port Switch 19” Rack • Host Adapters plus Cables
8	Racks: • Dell Poweredge 4210 • Rack Gabinet of 42U with Cables

6.2.2 ERC’s System Configuration

Similar to the CTC’s case, in year 2000, the ERC at MSU made the decision to change their core technology platform based on a dedicated system (Sun Enterprise 10000) to a commodity cluster (IBM and SGI Intel nodes with Linux operating system). On the next sections we will describe the configuration of the previous dedicated system and the new solution based on commodity cluster.

6.2.2.1 Previous System: Sun Enterprise 10000

The system that ran previously as the central provider for high performance computation at the ERC was a Sun Enterprise 10000 (E10000), a dedicated system from Sun Microsystems Corporation [78]. The E10000 was acquired by the ERC in 1998. The system was configured to its maximum capacity at the moment of purchase, as a consequence, it could only grow on power (upgrade of processors) but not on resources (number of processor and memory were to their limit). The following are some of the drawbacks that the ERC found if continuing using the E10000[74]:

- Flexibility of the system: The system was configured to its total capacity, like mentioned previously. The demands of high performance computation were increasing rapidly and a the system could not be flexible to meet these demands, being inefficient for the organization.
- High annual maintenance fee: The maintenance contract for the E10000 was very high, increasing the system's TCO.
- Cooling and power requirements: The E10000 required high power a cooling consumption, enough to equip at least three departmental systems. This also incremented system's TCO.

The configuration of the E10000 at the ERC is shown in Table 6.3 [74, 9].

Table 6.3 ERC's E10000 system configuration

Quantity	**Description**
1	Sun Enterprise 10000 System: • 64 UltraSPARC 333Mhz Processors • 64GB Memory RAM • Cross Bar Interconnection Switch of 1.2GB/sec.

The peak performance obtained from this configuration was 124 Gflops[2]. Although the ERC did not decommissioned the E10000 (which is still running for other purposes, and is a shared resource between the ERC and MSU), it is no longer the core for ERC's HPC demands.

6.2.2.2 The New Solution: IBM/SGI Lintel Cluster System

The ERC considered several alternatives to change their actual dedicated system to a more flexible and also cost efficient solution. The alternatives considered were [74]:

- Upgrade to a new version: At the moment of the evaluation, Sun Microsystems released the next version of the E10000, the Sun Fire 15000 (Sun Fire 15K). Sun Microsystems was offering to their E10000 based customers the opportunity of a program upgrade. This program consisted in receiving the old version (E10000) and installing the new version (Sun Fire 15K) with a better configuration at a fair price.
- Sun cluster: Instead of having a large dedicated system, ERC also considered to have several departmental Sun servers and interconnect them together for better performance.
- Intel network of workstations (NOW): The possibility to connect several commodity processors with a high bandwidth network was another alternative that the ERC was considering at the moment.

The upgrade to the Sun Fire 15K was discarded because of the maintenance contract cost of the system and the future limitation that the system will have when scaled up to its maximum capability (106 processors). The Sun cluster alternative was also discarded because of floor space issues. The Sun server nodes that were considered occupied more space (four rack units per node) that the amount desired by the ERC (two rack units per node).

[2]Reaching position #54 from the top 500 list in 1999

The commodity cluster system was the solution chosen by the ERC, with Intel-base nodes and Linux operating system. The system is called Empire (ERC's Massive Parallel Initiative for Research and Engineering) and has passed through several phases of configuration (four until present time). The Empire system was evaluated during Phase I and II of its configuration, shown in Table 6.4.

Table 6.4 ERC's Empire (Phase I & II) system configuration

Quantity	Description
132	**Phase I** Computational Nodes: IBM X330 Server • Dual 1GHz. Intel Pentium III Processors • 1GB Memory RAM • ≈ 54 GB of Internal Disk
4	Network Components: Extreme Network 48i 100Mbit/sec Switch
4	Rack Components: Rack Cabinet IBM NetBay (42U) with Cables
164	**Phase II** Computational Nodes: SGI 1100 Server • Dual 1GHz. Intel Pentium III Processors • 1GB Memory RAM • ≈ 54 GB of Internal Disk
5	Network Components: Extreme Network 48i 100Mbit/sec Switch
1	Extreme Network 7i Gigabit Ethernet Switch
5	Rack Components: Rack Cabinet SGI (42U) with Cables

This system reached a peak performance of more than 140 Gflops. The flexibility of this architecture had permitted the ERC to expand the cluster to more than 500 nodes

(more than 1024 processors), and be positioned among the 200 fastest computer systems in the world[3].

6.3 The Cost/Effective Metric Analysis

In this section, we will apply the cost/profit effective metric to the cases of the CTC and ERC, described in previous sections of this chapter. To have a better representation of the metric, we will model the proposed solution for the CTC and the ERC. These models will have a system configuration similar to the one seen on the new solutions (commodity clusters), with the technology available up to date. Then, we will determine the cost and profit of the models according to the $Cost(S,t)$ and $Profit(S,t)$ functions. Finally, we will discuss some analyses according to the results obtained.

6.3.1 Case 1: Velocity I, CTC's Proposed Solution

The CTC purchased the Velocity I cluster in August of 1999, at a total cost of approximately $3 million with a three year maintenance contract included in the price [55]. As previously discussed, this system can deliver a peak performance of 122 Gflops, obtaining approximately two to four Gflops per node. The Velocity I was purchased at one-fifth of the price of the previous system, the IBM SP2 (with 160 processors) [51, 55].

[3]according to the top 500 list of June 2002, with 366 Gflops of peak performance

Table 6.5 shows the configuration and price of a system [4] similar to the Velocity I configuration. This is our model system for the Velocity I, which we will call the VModel. The VModel system, as the Velocity I, also has a three year maintenance contract included in the purchase price.

Table 6.5 VModel system configuration

Quantity	Description	Price	Total
64	Computational Nodes: Dell Poweredge 6650 Server • Quad Processor Intel Xeon 1.6Ghz • 4MB Cache/Processor • 4GB Memory RAM • 54GB Internal Storage Disk	$25,938	$1,660,032
40	Network Components: • Emulex Xilan Switch GNX5000 • 8 Port Switch 19” Rack	$6,250	$250,000
128	• Host Adapters plus Cables	$795	$101,760
8	Racks: • Dell Poweredge 4210 • Rack Gabinet of 42U with Cables	$2,000	$16,000
		SubTotal	$2,027,792
		Discount (20%)	-$405,558
		Total	$1,622,234

Since the CTC is a non-profit organization, based on research grants for developing wide spectrum of applications, the profitability related to the technology is complex to obtain. As discussed in section 4.2 of chapter IV, another way to represent the profit of

[4]System configuration and prices were taken on August 2002, from Dell’s and Emulex’s website for the US

a system is by calculating its cost savings. In this case the $Profit(S,t)$ function of the Velocity I system can be obtained by the difference of the cost incurred by the previous system (IBM SP2) and the cost of the new system (Velocity I) in a year ($t = 1$). The annual cost of the IBM SP2 system (not counting the year of purchase) is based on only maintenance and operational costs (approximately $1 million annually [55]). Hence, the $Profit(\text{Velocity I}, 1)$ function results in a negative profit value for the first year of the Velocity I, as shown in Equation 6.1.

$$\begin{aligned} Profit(\text{Velocity I}, 1) &= Cost(\text{SP2}, 1) - Cost(\text{Velocity I}, 1) \\ Profit(\text{Velocity I}, 1) &\approx (-\$2M) \end{aligned} \tag{6.1}$$

The result for the first year was expected, since the purchase price of the hardware is included in the total cost (we are assuming complete purchase of the system and not a lease). For the next year, $t = 2$, we expect the profit to increment dramatically, since the maintenance cost of the Velocity I is included in the hardware cost (first year) and the remaining costs (operational) do not exceed 10% of the cost of the first year, being lower than the maintenance cost of the IBM SP2 in a year. This is shown in the graph of Figure 6.1.

We will now determine the $Profit(S,t)$ and the $Cost(S,t)$ functions for the VModel based on the configuration of Table 6.5. Equation 6.2 shows the representation of the VModel. This equation is analogous to the profit function for the Velocity I previously discussed.

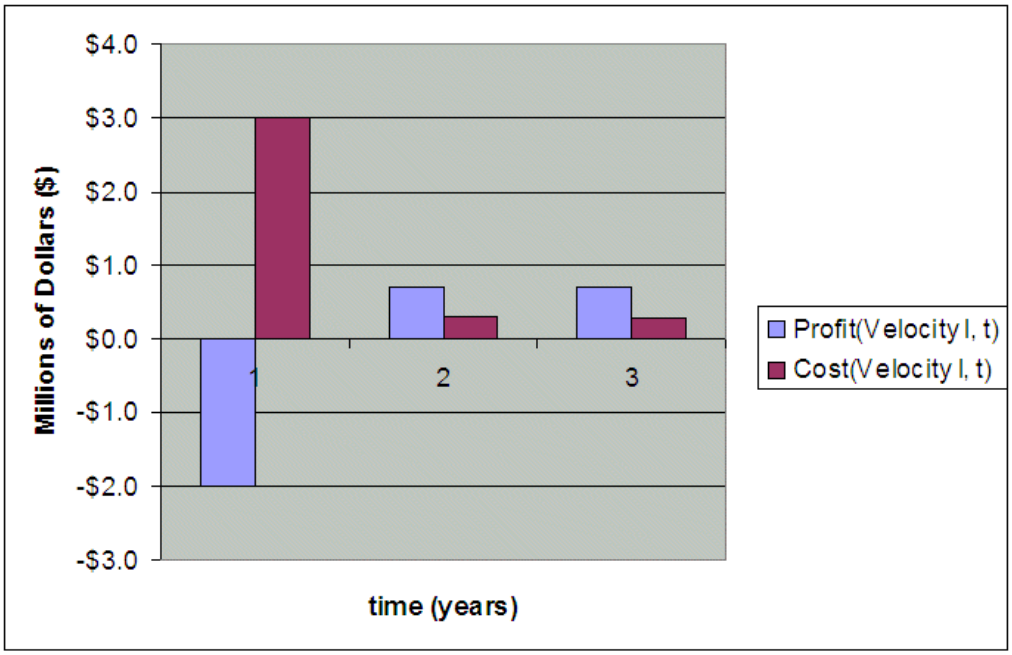

Figure 6.1 Cost and profit of the Velocity I in the first three years

$$Profit(\text{VModel}, 1) = Cost(\text{SP2}, 1) - Cost(\text{VModel}, 1) \tag{6.2}$$

Equation 6.3 shows the definition of the $Cost(S, t)$ function for the VModel annually ($t = 1$), according to chapter V.

$$Cost(\text{VModel}, 1) = \Sigma_c(\text{VModel}, 1) + \Theta_c(\text{VModel}, 1) \tag{6.3}$$

The explicit costs $\Sigma_c(\text{VModel}, 1)$ are shown in Equation 6.4. The hardware costs includes the price of the system from Table 6.5 and the hardware maintenance contract for three years. For the software costs of the VModel we will assume a cost of $1,000 per processor and an annual software maintenance contract of 20% of the cost of the software.

The utility cost is represented as 1% of the hardware cost annually. All these assumptions are according to the ranges discussed in section 5.1 in chapter V.

$$\begin{aligned}
&\Sigma_c(\text{VModel}, 1) = C_H(\text{VModel}, 1) + C_S(\text{VModel}, 1) + C_U(\text{VModel}, 1) \\
&C_H(\text{VModel}, 1) = C_A(\text{VModel}, 1) + f_d(\text{VModel}, 1) \approx \$2,054,826 \\
&C_S(\text{VModel}, 1) \approx \$281,600 \\
&C_U(\text{VModel}, 1) \approx \$16,222 \\
&\Sigma_c(\text{VModel}, 1) \approx \$2,352,648
\end{aligned} \tag{6.4}$$

The opportunity cost $\Theta_c(\text{VModel}, 1)$ is shown in Equation (6.5). We assumed that the system is being utilized 99.9998% of the time. Following Moore's law, we assume that the performance of the VModel system is approximately twice the performance obtained with the Velocity I, this is represented by $P(\text{VModel})$. We assumed no unpredicted downtime of the system (t_{unpred}) and a total of 96 hours annually for predicted downtime t_{pred}[5]. We also assume a unit gain rate $\varphi(P(\text{VModel}))$ of approximately $300 per hour [6].

$$\begin{aligned}
&\Theta_c(\text{VModel}, 1) = \rho_u(\text{VModel}, 1) + C_d(\text{VModel}, 1) \\
&\Theta_c(\text{VModel}, 1) = (1 - f_u) \times M + \varphi(P(\text{VModel})) \times (t_{pred} + t_{unpred}) \\
&\Theta_c(\text{VModel}, 1) = (1 - 0.999998) \times C_H(\text{VModel}, 1) + 100\$/hrs. \times 96hrs. \\
&\Theta_c(\text{Vmodel}, 1) \approx \$10,000
\end{aligned} \tag{6.5}$$

[5]Considering two hours weekly to perform maintenance and backup of the system

[6]Considering the total annual profit of approximately $850,000, similar to the profit obtained from the Velocity I in the secong year

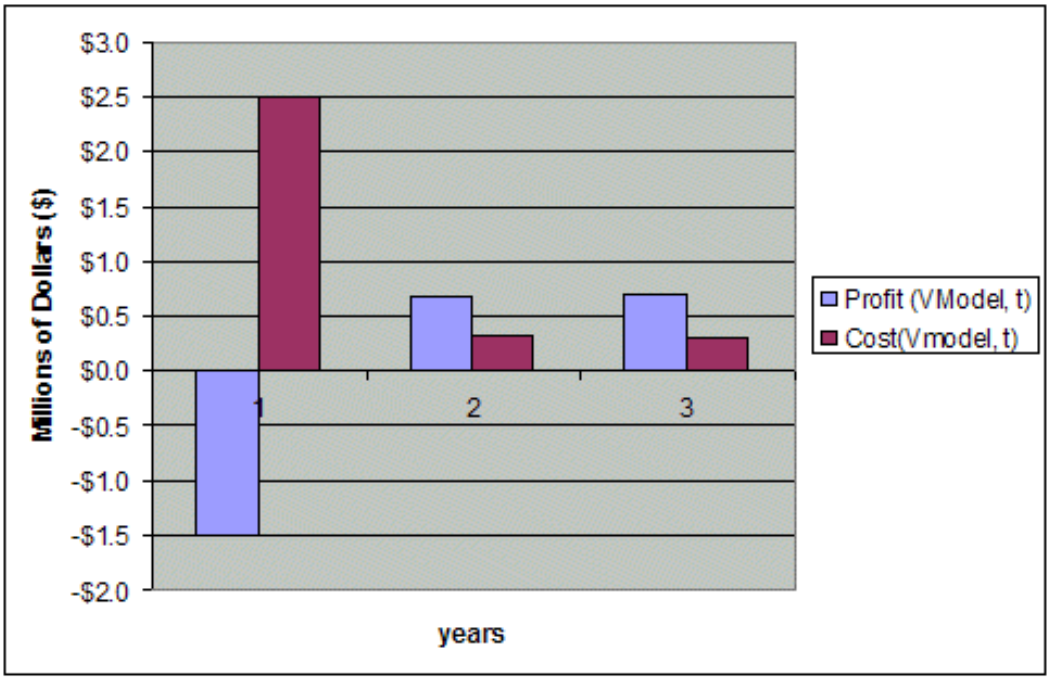

Figure 6.2 Cost and profit of the VModel in the first three years

Therefore, the total cost of the VModel system is approximately $2.5 million dollars, similar to the cost of the Velocity I 2 years ago. Figure 6.2 shows the behavior of the VModel in the first three years. As we can see, the behavior is similar to the behavior of the Velocity I shown in Figure 6.1. The VModel has better performance than the Velocity I system at approximately the same cost after two years, this fact is expected as a consequence from Moore's Law. Equation 6.6 represents the cost/profit effective metric applied to the VModel in the first year.

$$
\begin{aligned}
\Gamma_{profit}(\text{VModel}, 1) &= \frac{Profit(\text{VModel},1)}{Cost(\text{VModel},1)} \\
\Gamma_{profit}(\text{VModel}, 1) &= \frac{Cost(\text{SP2},1) - Cost(\text{VModel},1)}{Cost(\text{VModel},1)} \\
\Gamma_{profit}(\text{VModel}, 1) &= \frac{Cost(\text{SP2},1)}{Cost(\text{VModel},1)} - 1 \\
\Gamma_{factor}(\text{VModel}, 1) &= \frac{Cost(\text{SP2},1)}{Cost(\text{VModel},1)}
\end{aligned}
\tag{6.6}
$$

For the CTC case, if the IBM SP2 would have been upgraded to 256 nodes (between thin and wide nodes), and assuming that each node has an average cost of $50K (the smallest SP node is about $35K, assuming a strong discount [32]), the resulting cost of the upgrade would have been approximately $5 million, much higher than the cost reflected by the whole Velocity I system. Figure 6.3 shows the cost/profit effective metric for the Velocity I and the IBM SP2 systems.

In the next section we will discuss the ERC's case. We will go a step further by analyzing the cost/profit effective metric on one of the system alternatives considered by the ERC in comparison to the commodity cluster implemented.

6.3.2 Case 2: Empire (Phase I and II), ERC's New Solution

Previous to the actual commodity cluster system (EMPIRE), the ERC handled its heavy computational requests through a Sun E10000 system. This system with a configuration discussed in section 6.2.2, had a cost of approximately $1 million, and an annual maintenance cost of $500,000 [78, 74]. As mentioned, the system had to be replaced because it had reached its maximum capacity (could not scale-up). Among the systems proposed, we

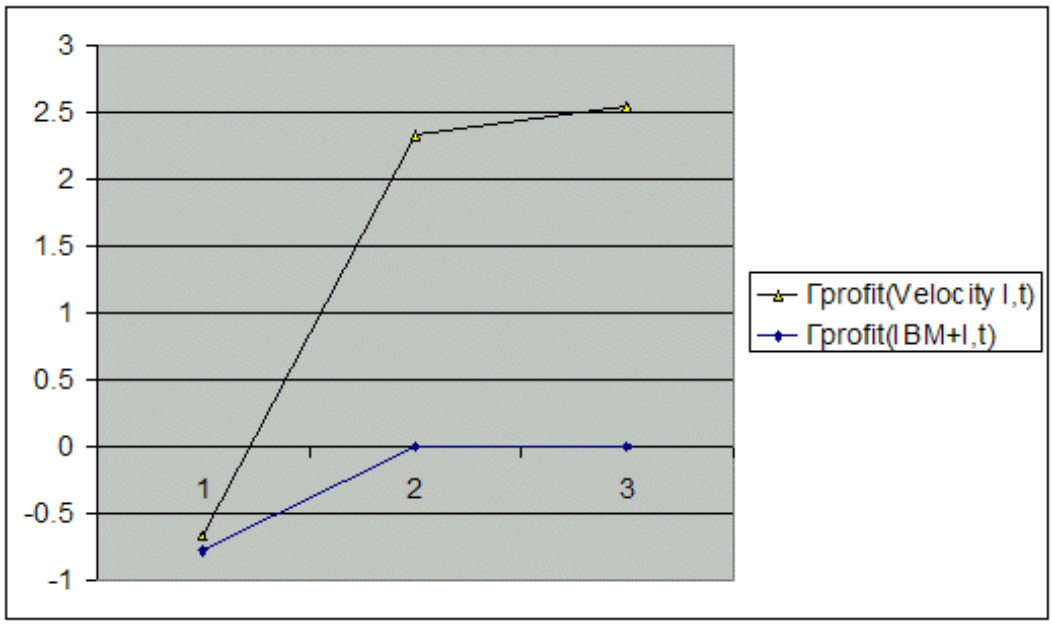

Figure 6.3 Cost/Profit Effective Metric: Velocity I versus IBM SP2 Upgrade

will analyze the Sun Fire 15K against the chosen system, the Intel-based cluster system (EMPIRE in configuration Phase I and Phase II), using the cost/profit effective metric.

In Table 6.6 we can observe the system configuration and price [7] for the Sun Fire 15K. The maintenance contract for a Sun Fire 15K hardware is $300,000 annually[8]. The software cost per processor is between $500 and $1,500, depending on the application. We will assume, $1,000 as we did previously with the CTC's case. We also maintain the depreciation rate at 3 years. Staff training is necessary for the operation of the system, costing approximately $40,000 annually[9]. Although utility costs are higher on dedicated systems,

[7]System configuration and prices were taken on August 2002 from Sun Microsystems web site (http://www.sun.com).

[8]Sun Microsystems HA Pack for Sun Fire 15K for one year.

[9]Sun Microsystems educational skills package for Sun Fire 15K for one year.

Table 6.6 Sun Fire 15K system configuration and price

Quantity	Description	Price	Total
1	Sun Fire 15K • 72 Processors UltraSPARC 3 1.05Ghz • 8MB Cache/Processor • 288GB Memory RAM • 18x18 Crossbar Switch (172GB/sec) • 72GB Internal Disk • Sun Solaris 8.0	$3,740,135	$3,740,135
		SubTotal	$3,740,135
		Discount (20%)	-$748,027
		Total	$2,992,108

due to power consumption, we will also maintain the utility cost at 1% of the hardware cost, annually. Then, the explicit costs of the Sun Fire 15K, annually, are presented as,

$$
\begin{aligned}
&\Sigma_c(\text{Sun15K}, 1) = C_H(\text{Sun15K}, 1) + C_S(\text{Sun15K}, 1) + C_U(\text{Sun15K}, 1)\\
&C_H(\text{Sun15K}, 1) = C_A(\text{Sun15K}, 1) + f_d(\text{Sun15K}, 1) \approx \$4,000,000\\
&C_S(\text{Sun15K}, 1) \approx \$100,000\\
&C_S u(\text{Sun15K}, 1) \approx \$340,000\\
&C_U(\text{Sun15K}, 1) \approx \$40,000\\
&\Sigma_c(\text{Sun15K}, 1) \approx \$4,480,000 (< \$4.5M).
\end{aligned}
\tag{6.7}
$$

To simplify our analysis we will assume no opportunity costs ($\Theta_c(\text{Sun15K}, 1) = 0$), with zero system downtime[10] and utilization of 100%. As a result, Figure 6.4 shows the cost and profit of the Sun Fire E15K for the first three years.

[10]Sun E10000 and Sun Fire 15K are based on dynamic reconfiguration which permits to do maintenance of the system without bringing the system down.

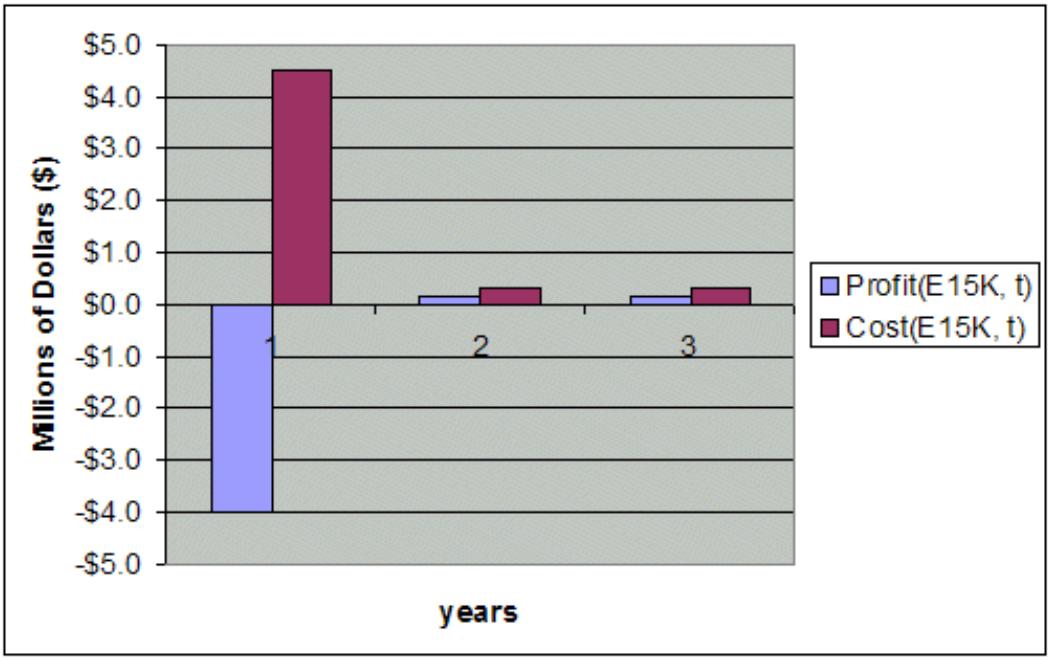

Figure 6.4 Cost and profit of the Sun Fire E15K in the first three years

Now we will analyze the configuration of the Empire system (phase I and II), chosen by the ERC. In Table 6.7 we find the configuration and price[11] of the Empire. The maintenance contract is included in the hardware price for a three years period. Again, the software cost per processor is $1,000, depreciation of the system is three years, and utility cost is 1% of the hardware cost. The explicit costs of the Empire (phase I and II), annually, are presented in Equation 6.8.

[11]System configuration and prices were taken on August 2002 from IBM, SGI, and Extreme Network websites.

Table 6.7 Empire system configuration and price

Quantity	Description	Price	Total
	Phase I		
132	Computational Nodes: IBM X330 (1U) Server • Dual 1GHz. Intel Pentium III Processors • 1GB Memory RAM • ≈ 54 GB of Internal Disk	$4,721	$624,492
4	Network Components: Extreme Network 48i 100Mbit/sec Switch	$8,000	$32,000
4	Rack Components: Rack Cabinet IBM NetBay (42U) with Cables	$1,539	$6,156
	Phase II		
164	Computational Nodes: SGI 1100 Server • Dual 1GHz. Intel Pentium III Processors • 1GB Memory RAM • ≈ 54 GB of Internal Disk	$4,495	$737,180
5	Network Components: Extreme Network 48i 100Mbit/sec Switch	$8,000	$40,000
1	Extreme Network 7i Gigabit Ethernet Switch	$70,000	$70,000
5	Rack Components: Rack Cabinet SGI (42U) with Cables	$1,800	$9,000
		SubTotal	$1,518,828
		Discount (20%)	-$303,766
		Total	$1,215,062

$$
\begin{aligned}
&\Sigma_c(\text{Empire}, 1) = C_H(\text{Empire}, 1) + C_S(\text{Empire}, 1) + C_U(\text{Empire}, 1) \\
&C_H(\text{Empire}, 1) = C_A(\text{Empire}, 1) + f_d(\text{Empire}, 1) \approx \$2,000,000 \\
&C_S(\text{Empire}, 1) \approx \$100,000 \\
&C_S u(\text{Empire}, 1) \approx \$0 \\
&C_U(\text{Empire}, 1) \approx \$20,000 \\
&\Sigma_c(\text{Empire}, 1) \approx \$2,120,000 (< \$2.5M)
\end{aligned}
\tag{6.8}
$$

As we did with the Sun Fire 15K, we will assume zero system downtime and a full utilization of the system, this will set equal grounds for both comparisons. Therefore, Figure 6.5 describes the cost and profit of the Empire (phase I and II) system for the first three years.

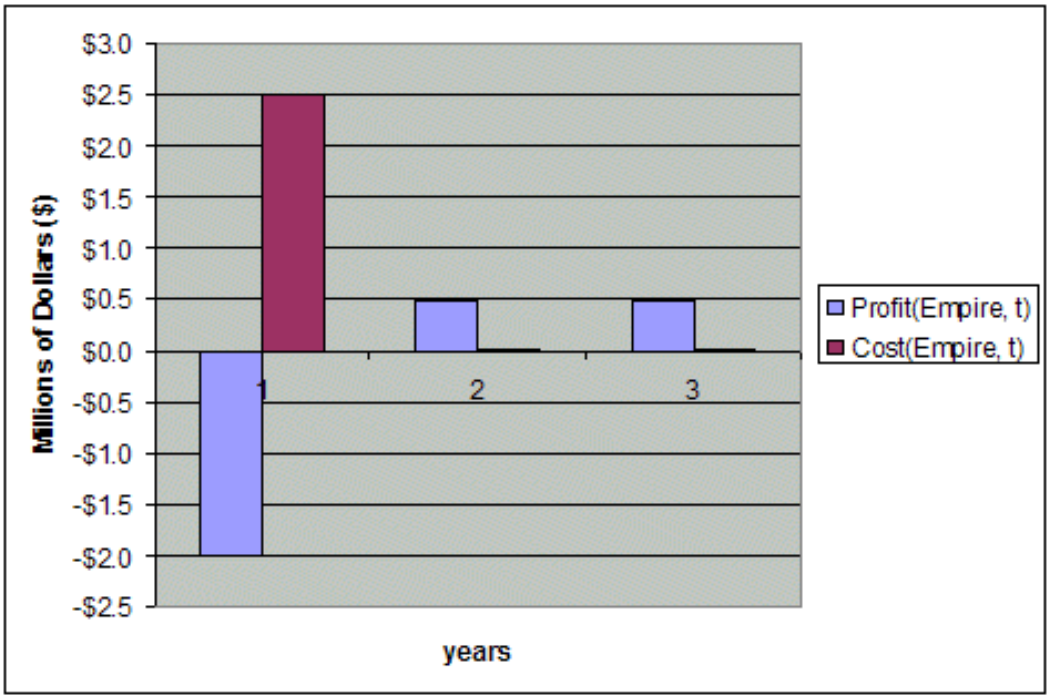

Figure 6.5 Cost and profit of the Empire (I and II) in the first three years

We can now look at the representation of the cost/effective metric for each of the systems in the first three years, as shown in Figure 6.6. Here we can see how the profit (cost savings in this case) for the Empire (phase I and II) is higher than the alternative Sun Fire 15K. This was an expected result since, as opposed to the Sun Fire 15K, the Empire system hardware maintenance cost is already included in the hardware cost. This same scenario was observed in our previous example for the CTC case. In general, commodity clusters tend to reduce cost of ownership by including maintenance contracts in the purchase price of the system. This is not the case in dedicated systems since the highly specialized hardware requires separate maintenance contracts in the range of thousands of dollars annually, thus increasing cost of ownership.

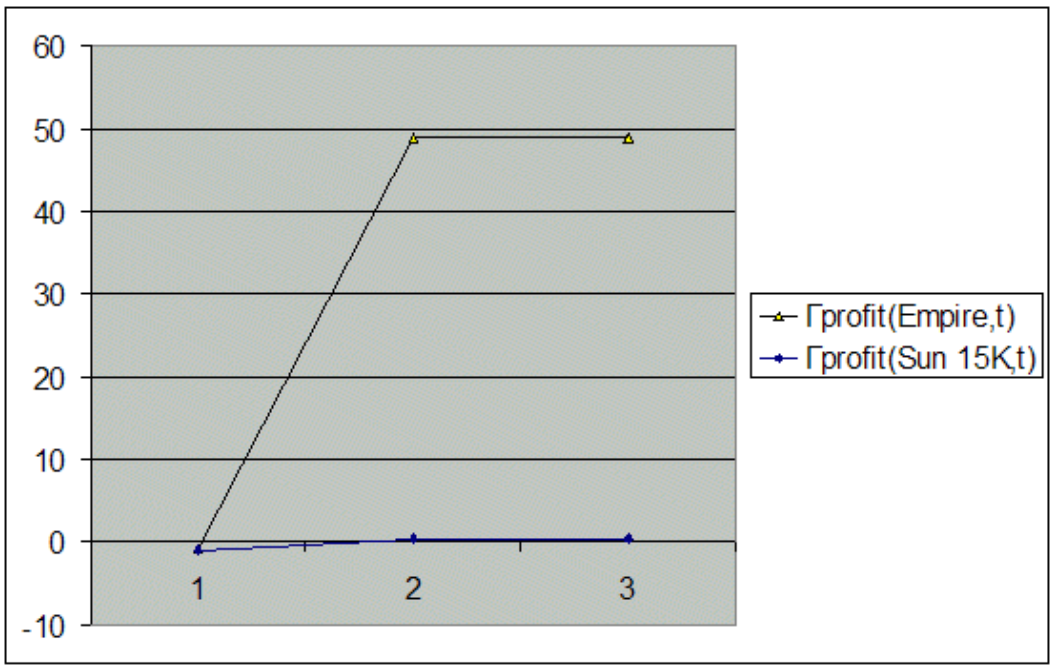

Figure 6.6 Cost/Profit Effective Metric: Empire (I and II) versus Sun Fire 15K

For the ERC, the decision of implementing the Empire (phase I and II) was obvious considering the other alternatives. The Empire system is able to reach a peak performance of 180 Gflops [74] with a cost lower than its counterpart, the Sun Fire 15K, which provides a peak performance of 190 Gflops. Currently, due the flexibility of the Empire, phase III and IV have been implemented, expanding the system to 1038 processors, thus reaching a peak performance of 366 Gflops[12], with a cost that still is below the cost of the Sun Fire 15K.

For both cases presented, the selection of the systems was influenced by the economic advantages offered by commodity clusters. Although there exists a trend to implement commodity clusters, this is not always the case. In some scenarios, the implementation of a cluster is not obvious (not the best) due to application performance, *e.g.* if the application scales-up poorly, there may be large memory requirements not feasible on distributed systems such as clusters. In these cases, the alternative is biased toward dedicated SMP architectures, even though their prices are not as competitive as the price of commodity clusters. However, this can be balanced if the application has a direct impact on the business profit.

[12]Reaching position #127 on the Top 500 List of June 2002

CHAPTER VII

CONCLUSION

The application of high performance computing (HPC) is no longer limited to government and academic research, it has expanded in recent years to other industry areas (financial, manufacture, telecommunications, oil and gas, etc.). Traditional systems that deployed high performance computing in the past were built using expensive resources of considerable size that needed special accommodation (they could occupy a whole data center). A staff of many high skilled personnel (up to hundreds) was also necessary to manage and operate this type of systems. At the time, only government and solid companies were able to afford HPC.

Technological advances permitted a downsizing of traditional HPC systems. A variety of vendors emerged (*e.g.*, IBM, HP, Sun, etc.), offering proprietary architectures and operating systems for HPC. Recently, a new wave of delivering HPC has been implemented using networks of commodity processors (NOW, COW, etc.), moving the proprietary features of parallel systems to a more open and flexible architecture.

The diversity of configurations available nowadays has made it difficult for information technology (IT) managers to evaluate their options from a point of view of economic and performance benefits. Moreover, IT is playing an important role in many businesses.

Justification of HPC investment is not an option anymore. In this thesis, we presented a framework for evaluating parallel systems in terms of performance, cost and profit. We introduced a metric from which results were analyzed to support the economic justification for purchasing a parallel system in terms of cost and profit.

We discussed previous work for evaluating parallel systems. We found that many of the evaluations were completed by measurements based on the performance of the system and very few included cost and profit implications. We also found some economic models for IT systems such as: Total Cost of Ownership (TCO), Total Economic Impact (TEI) and others. These models only focused on the cost of operating a parallel system but did not considered the return on investment. A categorization of parallel architectures was also studied to understand the factors that are considered when evaluating of parallel systems, such as: shared memory systems (SMP), and distributed memory systems (MPP and Clusters).

We then developed the framework by first identifying the relevant factors that should be considered when evaluating parallel systems. These factors are grouped into four categories: potential, financial, environmental and cultural. We then defined the effectiveness of a parallel system in terms of cost and profit, stating that a parallel system is cost/profit effective if the profit obtained from the system performance is greater than the total cost of the system in a period of time. We represented a measurement for this definition by the cost/profit effective metric. This metric was defined as the ratio of two functions: profit and cost.

The profit function was based on the profit (in dollars) obtained from the performance of a parallel system configuration in a period of time. We introduced the economic definition of production function to model the way the performance of a parallel system can affect the business profit. According to the business model we found that the measurement for system performance varies. We generalized this situation by classifying the performance of a parallel system in three cases: speed-up, throughput and scale-up. In cases where the profit cannot be based on system performance we discussed other benefits that can also impact the business.

The cost function was based on the total cost of a parallel system during its life cycle. We divided the cost function into explicit and opportunity costs. The explicit costs considered all the costs incurred in the purchase process: hardware, software, maintenance, and support. Opportunity costs were identified by the costs of downtime and underutilization of the system.

To analyze the usage of the cost/profit effective metric we applied it to two study cases: the Cornell Theory Center (CTC) and the Engineering Research Center (ERC). These two high performance computational centers had made a transition from dedicated systems to a cluster system. We modeled both scenarios and applied our framework to the old and new configurations calculating the profit and cost functions. The cost/profit effective metric allowed us to express the advantages of the new configuration by using quantitative estimates expressed in business terms (dollars). Although there exists a trend to implement commodity clusters, this is not always the case. Applying the cost/profit effective metric

can aid to determine and quantify in detail the relevant factors for deciding to replace or upgrade a parallel system in terms of performance, cost and profit. In general, it is possible to apply this quantitative analysis to many scenarios using the ERC and CTC examples as a guide.

7.1 Strength of this Thesis

It is important to note that although many metrics have been proposed in order to evaluate parallel systems, we believe that the framework introduced in this thesis is well suited for the cases of evaluating parallel systems in terms of performance, cost and profit. Our main focus was on the scenario of deciding between the replacement or upgrade of a parallel systems. We chose this situation because it presented the majority of cases where an IT manager had to evaluate in detail a parallel system.

The evaluation of a parallel systems based on its purchase cost is not enough. The idea of "performance at any cost" is a costly solution. IT managers are now more concerned with the tradeoffs between performance and cost, and need to find ways to justify their decisions to management executives. Many IT managers will benefit from the framework developed in this work. For example, in an e-business organization an IT manager will be able to analyze if investing on a new system will bring more profit to the business than expanding the existing infrastructure. Many IT managers do not feel confident when implementing changes to their current platform because of the risks associated with those changes (delay on production, users not performing efficiently with the new imple-

mentation, system's cost of ownership surpassing the expected budget, etc.). With this framework, IT managers are able to identify and quantify these risks beforehand, at the beginning of any system project. Besides, it is possible to have a quantifiable presentation of the project for the high management executives.

Although, different analysis have been proposed in this area (performance analysis, scalability analysis, TCO, etc.), only a few combine the evaluation of parallel systems in terms of the performance and economic factors that can impact the business goals. The cost-effective metric [85] and the profit-effective metric [86] being the most relevant work found in our research. Our work is the only work (to the best of our knowledge) that applies a metric that combines performance, cost and profit factors to analyze the parallel system effectiveness from the business point of view. Other metrics have focused mainly on the performance or the cost analysis but not on both. The main contribution of our work is to provide an unified view of all the studies and metrics used to evaluate parallel systems in terms of performance, cost and profit.

To conclude, our main contribution to this area of research is twofold: first, create a framework to identify all the factors that are present when evaluating a parallel system, focusing on the situation of replacing versus upgrading, and second, the development of a metric that combines the factors of performance, cost and profit found in this framework. The application of this metric in two real life scenarios also demonstrated how this metric can be applied.

7.2 Lessons Learned

During the elaboration of this thesis, we came across a set of issues that, due to time constraints, were excluded from this work. Most of this issues were presented in the application of the metric to non profit organizations, where the profit function had to be determined by other benefits besides the production function described in Chapter IV. Although, the way that profit was represented in this case (cost savings) is a valid way to identify the benefits of a system, we believe this can be improved by applying it to profit organizations where the performance of a systems plays a main role in the business. In general, identifying the profit for organizations that are not service-driven or technology-centric (e-business, telecommunications, etc.) is a difficult task.

As time goes by, organizations will demand more analysis of IT infrastructure and the overall impact on the business. The tendency of HPC to transition to commodity clusters is expanding and threatening many traditional system vendors with open system architectures. Still, with this new wave, organizations prefer to wait until these trends mature and keep relying the implementations of their core applications on traditional systems and support from system vendors.

We have also learned that great part of evaluating parallel systems depends on the application we want to implement. For example many applications are not well suited for distributed memory architecture. Many commercial applications have already been ported to distributed memory architecture, but there is still a majority of applications requiring shared memory (SMP, NUMA, etc.). This fact can often constraint IT managers to acquire

expensive parallel systems. Other type of applications that can influence the evaluation of parallel systems are the implementation of queuing and operating systems. Poor implementations can cause the system to function inefficiently altering the costs of operation (resources wasted or underutilization). It is important to consider all these factors when evaluating parallel systems as we mentioned in this work.

7.3 Future Work

Evaluating parallel systems in terms of performance, cost and profit is a very broad research area. Our focus was primarily on deciding between upgrading versus replacing a system, as mentioned before, because is the most common situation where IT managers evaluate parallel systems.

In our study case we were able to show how to apply the cost/profit effective metric in a situation where replacing a dedicated system to a commodity cluster was required. The examples were cases of non-profit organizations where the profit was obtained solely on cost savings and the analysis for replacement was obvious due to the benefits that commodity clusters provide. It will be interesting to expand this test to organizations where the profit can be related to system performance. In this case we will have to determine the unit gain rate $\varphi(P(S_i))$ described in Chapter IV, and the performance unit by which the system is evaluated according to the business model: speed-up, throughput or scale-up.

We also have to notice that in our case example we considered the total purchase of the systems. In our cost/profit effective metric there is a factor that corresponds to the

system depreciation. It will be interesting to also add a factor of amortization for the cases where leasing is considered. This type of change can expand the usage of the metric to more general cases work more realistic and accurate evaluations. We strongly encourage the application of this framework to wider range of domains besides the examples shown here. For example in service-driven and process centric organizations where the financial justification on IT investment is crucial for the business.

REFERENCES

[1] B. Adams, *Small Business Start-Up*, Adams Media Corporation, 2001.

[2] S. Allamaraju, "Nuts and Bolts of Transaction Processing," http://www.subrahmanyam.com/articles/transactions/, May 1999.

[3] G. M. Amdahl, "Validity of Single-Processor Approach To Achieving Large-Scale Computing Capability," *Proceedings of American Federation of Information Processing Societies (AFIPS) Conference*, Reston, Virginia, 1967, pp. 483–485.

[4] E. Anderson, Z. Bai, C. Bischop, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, Tech. Rep., Society for Industrial and Applied Mathematics, Philadelphia, Philadelphia, 1995.

[5] D. Bailey et al., *The NAS Parallel Benchmarks*, Tech. Rep., NASA Ames Research Center, March 1994.

[6] D. J. Becker, T. Sterling, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer, "Beowulf: A Parallel Workstation for Scientific Computation," *Proceedings of the 1995 International Conference on Parallel Processing*, vol. 1, 1995.

[7] L. S. Blackford et al., *ScaLAPACK User's Guide*, Siam Publishing, 1997.

[8] R. Brightwell and S. J. Plimpton, "Scalability and Performance of Two Large Linux Clusters," *Journal of Parallel and Distributed Computing*, vol. 61, 2001, pp. 1546–1569.

[9] E. R. Center, "Facilities and Equipment of the ERC Center," http://www.erc.msstate.edu/about/facilities/, Accessed 09/20/2002.

[10] Center for Parallel Computing Research, "Parallel Computing Research Newsletter," October 1993, Volume 1, Issue 4.

[11] Compaq Computer Corporation, "Achieving Business Value by Measuring and Managing the Cost of Computing," White Paper, March 1998, First Edition.

[12] P. Computation, "Frontier: The Premier Internet Computing Platform," http://www.parabon.com/clients/internetComputingWhitePaper3.jsp, Accessed: 09/18/2002.

[13] S. Cook and R. Reckhow, "Time Bounded Random Access Machine," *Journal of Computer and System Sciences*, vol. 7, no. 4, August 1973, pp. 354–375.

[14] "Cornell Theory Center," http://www.tc.cornell.edu, Accessed 09/14/2002.

[15] Cornell Theory Center, "AC3 Velocity Up and Running in Less than a Day," http://www.tc.cornell.edu/news/Events/1999/VelocityInstall/, Accessed 09/30/2002.

[16] Cornell Theory Center, *Parallel Performance Considerations*, Tech. Rep., Center for Parallel Computers, KTH, http://www.pdc.kth.se/training/Talks/Performance/ParallelPerfCons/more.html, 1999, Accessed 09/12/2002.

[17] "CTC High Performance Solutions," http://www.ctc-hpc.com, Accessed 09/14/2002.

[18] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eiken, "LogP: Towards a Realistic Model of Parallel Computation," *Procedings of the 4th ACM Symposium on Principles and Practice of Parallel Programming (PPOPP)*, San Diego, California, May 1993, pp. 1–12, ACM Press.

[19] A. Davies, *Computational Intermediation: A Model for Pricing Internet Distributed Computation*, Tech. Rep., Duqesne University, Pittsburgh, Pennsylvania, 2002.

[20] B. Day and J. Erickson, "Total Economic Impact Analysis: Oracle 9i Real Application Clusters on Compaq Trucluster Server Software," Giga Information Group White Paper, December 2001.

[21] W. J. Doherty, "Scheduling TSS/360 for Responsiveness," *Procedings of the AFIPS FJCC*, 1970, pp. 97–111.

[22] J. J. Dongarra, *Performance of Various Computers Using Standard Linear Equations Software, (Linpack Benchmark Report)*, Tech. Rep., University of Tennessee Computer Science, 2002.

[23] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK Users' Guide*, 1979.

[24] D. L. Eager, J. Zahorjan, and E. D. Lazowska, "Speedup Versus Efficiency in Parallel Systems," *IEEE Transactions on Computers*, vol. 38, no. 3, March 1989, pp. 408–423.

[25] "Engineering Research Center (ERC)," http://www.erc.msstate.edu, Accessed 09/20/2002.

[26] EUROPORT-D, "Computer Simulation Reduces Need for Prototype Testing in Car Industry," http://www.gmd.de/SCAI/europort-d/restraint-systems1.html, Accessed 10/08/2002.

[27] EUROPORT-D, "Polymer extruxion problem," http://www.gmd.de/SCAI/europort-d/polymer-extrusion.html, Accessed 09/12/2002.

[28] EUROPORT-D, "Typical CFD problems in car design," http://www.gmd.de/SCAI/europort-d/fluid-dynamics1.html, Accessed 09/12/2002.

[29] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," *IEEE Transactions on Computers*, vol. C-21, no. 9, September 1972, pp. 948–960.

[30] S. Fortune and J. Wyllie, "Parallelism in Random Access Machines," *Conference Record of the 10th Annual ACM Symposium on Theory of Computing*, San Diego, California, May 1978, pp. 114–118.

[31] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, *PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing*, 1994.

[32] W. Goedicke, "Evaluating the Relative Merits of IBM's SP System," http://acs.neu.edu/cs-tech/marginalia/SP_eval/, December 1998, Accessed 09/20/2002.

[33] R. M. Graham, *Advanced Course on Software Engineering*, chapter Performance Prediction, Springer-Verlag, 1973, pp. 395–463.

[34] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, chapter 4, The Benjamin/Cummings Publishing Company Inc., Redwood City, California, 1994, pp. 117–147.

[35] A. Grama, A. Gupta, and V. Kumar, *Isoefficiency Function: A Scalability Metric for Parallel Algorithms and Architectures*, Tech. Rep., University of Minnesota, 1993.

[36] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, San Mateo, California, 1993.

[37] A. Gupta and V. Kumar, "Analyzing Scalability of Parallel Algorithms and Architectures," *Proceedings of the 26th Hawaii International Conference on System Science*, Hawaii, 1993, IEEE Computer Society, pp. 144–153.

[38] A. Gupta and V. Kumar, "Performance Properties of Large Scale Parallel Systems," *parallel and distributed computing*, vol. 19, 1993, pp. 234–244.

[39] J. L. Gustafson, "Reevaluating Amdahl's Law," *Communications of the ACM (CACM)*, vol. 31, no. 5, May 1998, pp. 532–533.

[40] D. Heller, "Ten Steps for Managing Parallel Computing Projects," *IEEE Parallel and Distributed Technology*, Spring 1994, pp. 6–8.

[41] M. D. Hill, "What is Scalability?," *Computer Architecture News*, December 1990, pp. 18–21.

[42] S. A. Hodge, "Tax Law Meets Moore's Law," http://www.taxfoundation.org/foundationmessage10-00.html, October 2002, Foundation Message.

[43] L. Hu and I. Gorton, "Performance Evaluation for Parallel Systems: A Survey," October 1997.

[44] "IBM," http://www.ibm.com, Accessed 09/20/2002.

[45] "International Data Corporation (IDC)," http://www.idc.com, Accessed 09/18/2002.

[46] "Intel Corporation," http://www.intel.com, Accessed 09/18/2002.

[47] R. Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

[48] D. Jones, "The True Cost of Computing," *ESTAIC Newsletter*, , no. 1, June 1998.

[49] V. Kumar and A. Gupta, "Analyzing the Scalability of Parallel Algorithms and Architectures," *Parallel and Distributed Computing*, 1994.

[50] P. M. Lewis, A. Bernstein, and M. Kifer, *Databases and Transaction Processing: An Application-Oriented Approach*, Addison Wesley, Boston, Massachusetts, 2002.

[51] D. A. Lifka, *High Performance Computing with Microsoft Windows 2000*, Tech. Rep., Cornell Theory Center, Ithaca, New York, 2001.

[52] E. Luke, I. Banicescu, and J. Li, "The Optimal Effectiveness Metric for Parallel Application Analysis," *Information Processing Letters*, vol. 66, no. 5, June 1998, pp. 223–229.

[53] N. Magazine, "Realizing ROI on IT," http://www.networkmagazineindia.com/200207/index.shtml, July 2002, Accessed 09/08/2002.

[54] T. Malthus, *An Essay on the Principle of Population*, J. Johnson, London, England, 1978.

[55] J. Mayer, "Off-the-shelf Clusters Give Supercomputers a Run for Their Money," http://www.itworld.com/Comp/1437/ITW001121supercomputers/, November 2001, Accessed 09/20/2002.

[56] R. McCain, *Essential Principles of Economics: A Hypermedia Text*, chapter Chapter 7, Productivity and Supply, Electronic Book. Drexel University, http://william-king.www.drexel.edu/top/prin/txt/EcoToC.html, 2002, Accessed 09/08/2002.

[57] G. E. Moore, "Cramming More Components Onto Integrated Circuits," *Electronics*, vol. 38, no. 8, 1965, pp. 114–117.

[58] S. A. Morton, J. R. Davis, H. L. Duffey, G. L. Donathan, V. Forsyth, and S. N. Checkles, *Industrial Sismic Imaging on Commodity Supercomputers*, Tech. Rep., Amerada Hess Corporation, September 1999.

[59] MPI Forum, "MPI: A Message Passing Interface Standard," International Journal of Supercomputer Applications and High Performance Computing.

[60] T. R. Norton, "End-To-End Response Time: Where to Measure?," *Proceedings of 1999 Computer Measurement Group Conference (CMG99)*, Reno, Nevada, December 1999.

[61] T. R. Norton, "End-To-End Scaling: The Response Time Pipe," *CMG2001 Session 3208*, http://www.simalytic.com/CMG01/3208pptHO.pdf, December 2001.

[62] D. O'Connell, "TCO: Don't Let Hidden IT Expenses Hurt Your Company," *Software Magazine*, August 1998.

[63] G. Papadopolous, *The Future of Scalable Systems: The Interplay of Architecture and Management*, Tech. Rep., Sun Microsystems, http://ipdps.eece.unm.edu/1998/papers/papa.pdf, 1998.

[64] J.-P. Patino-Martnez, B. Ketotoe, and G. Alonso, *Improving the Scalability of Fault-Tolerant Database Clusters*, Tech. Rep., Technical University of Madrid, 2002.

[65] "Typical Power Consumption [Watt]," http://www.macinfo.de/hardware/strom.html, Accessed 09/18/2002.

[66] P. Redfern, "Conference call with Cornell Theory Center," September 2002, Call taken at ERC room 10 on the 17th from 1 to 2:00pm.

[67] D. Reed, "Accounting in the Age of Moore's Law," *CONTEXT Magazine*, Summer 1998.

[68] S/390 Division, *Industry Leadership for e-Business: e-Transaction Processing*, Tech. Rep., IBM, http://www-1.ibm.com/servers/eserver/zseries/library/whitepapers/gf225100.html, May 1999, Accessed 09/18/2002.

[69] "Sandia National Laboratories," http://www.sandia.gov/, Accessed 09/18/2002.

[70] R. Schenk, "The Production Function," http://ingrimayne.saintjoe.edu/econ/ The-Firm/ProductionFunct.html, 1998, Accessed 09/07/2002.

[71] Y. Shi, *Reevaluating Amdahl's Law and Gustafson's Law*, Tech. Rep., CIS Department of Temple University, Philadelphia, Philadelphia, October 1996.

[72] H. J. Siegel, S. Abraham, et al., "Report of the Purdue Workshop on Grand Challenges in Computer Architecture for the Support of High Performance Computing," *Parallel and Distributed Computing*, vol. 16, 1992, pp. 199–211.

[73] A. Silberschatz, H. K. Korth, and S. Sudarshan, *Database System Concepts*, 3rd. edition, McGraw-Hill, Boston, Massachusetts, 1999.

[74] R. Smith, "Interview with the Engineering Research Center (ERC)," January 2002, roger@ERC.msstate.edu.

[75] R. Standish, "SMP vs. Vector: A Head-to-head Comparison," *Proceedings of the HPCAsia 2001*, 2001.

[76] J. E. Stiglitz, *Principles of Microeconomics*, W.W. Norton and Company, New York, New York, June 1997.

[77] P. A. Strassmann, "IT Paradox Number," *Computerworld*, May 1999.

[78] "Sun Microsystems," http://www.sun.com, Accessed 09/20/2002.

[79] Techwise Research, "The Total Cost of Ownership for Enterprise Clusters," January 2002, Version 2.0.

[80] "Transaction Processing Performance Council (TPC)," http://www.tpc.org, Accessed 09/05/2002.

[81] UniFrame, "QoS Catalog for Software Components," http://magellan.cs.iupui.edu:8080/qoscat/index.htm, Accessed 09/18/2002.

[82] L. G. Valiant, "A Bridging Model for Parallel Computation," *CACM*, vol. 33, no. 8, August 1990, pp. 103–111.

[83] A. van der Steen and J. Dongarra, *Handbook of Massive Data Sets*, chapter Overview of High Performance Computers, Kluwer Academic Publishers, 2001.

[84] "Webopedia," http://www.webopedia.com, Accessed 09/15/2002.

[85] D. Wood and M. Hill, "Cost-Effective Parallel Computing," *IEEE Computer*, vol. 28, no. 2, February 1995, pp. 69–72.

[86] Y. Yan and X. Zhang, "Profit Effective Parallel Computing," *IEEE Concurrency*, vol. 7, no. 2, April 1999, pp. 65–69.