Fall 2006

Performance Benchmarks for Custom Applications: Considerations and Strategies

Braulio J. Cabral
Regis University

Regis University
School for Professional Studies Graduate Programs
Final Project/Thesis

Disclaimer

PERFORMANCE BENCHMARKS FOR CUSTOM APPLICATIONS: CONSIDERATIONS AND STRATEGIES

Braulio J. Cabral

Regis University

School for Professional Studies

Master of Science in Computer Information Technology

PERFORMANCE BENCHMARKS FOR CUSTOM APPLICATIONS: CONSIDERATIONS AND STRATEGIES

Braulio J. Cabral

Donald E. Archer, Degree Chair

Dr. Douglas Hart, Advisor

Computer Information Technologies

Abstract

The motivation for this research came from the need to solve a problem affecting not only the company used in this study, but also the many other companies in the information technology industry having similar problem: how to conduct performance benchmarks for custom applications in an effective, unbiased, and accurate manner. This paper presents the pros and cons of existing benchmark methodologies. It proposes a combination of the best characteristics of these benchmarks into a methodology that addresses the problem from an application perspective considering the overall synergy between operating system and software. The author also discusses a software design to implement the proposed methodology. The methodology proposed is generic enough to be adapted to any particular application performance-benchmarking situation.

Acknowledgements

I would like to express my gratitude to all those who provided me with their valuable support to complete this thesis. I want to thank my company for allowing me to begin this thesis and use the company resources and data for my research, I am indebted to my project advisor Dr. Douglas Hart from Regis University for taking time to review several drafts of this paper and provide constructive criticism and suggestions. Also thanks to my friend Army Sergeant Major Dwayne Forquer for taking time to listen to my arguments about this work.

I am especially obliged to Professor Suzanne Mannes, Ph.D. from Mannes Editorial for helping me with the big task of editing my work and to my fellow student Heather Baumgartner; Heather you did an excellent job on the peer review of this paper, thanks. Especial thanks to my wife Walky for proof reading the final draft of this paper. Also thanks to my wonderful kids Erik, Caitlyn, Braulio, Aibblyn and Biki for all the hours I took away from them while working on this paper. Of course, I have to thank my mother-in-law Elena for making sure I had plenty of coffee to drink during my work. Finally, I would like to thank Almighty God for giving me the strength to accomplish those things I put my mind into.

CONTENTS

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Introduction

With the fast growth of the global economy, Internet, organizations are forced to adopt real-time and event-driven data processing strategies and to rely less on batch processing for their day-to-day data interchange needs, specially in industries such as financing, medical, and supply chain (Freedman, 2006; Mohamed, 2006). Real-time data processing imposes a big burden on companies, forcing them to utilize state-of-the-art hardware and software in order to keep up with customers' demands (Ranadivé, 1999). It is important to understand customers' requirements and be able to accommodate products and services to meet these demands.

Implementing a successful real-time, event-driven data processing system requires understanding the potential workload and system capacity to process the workload in order to minimize latency (Gertphol, 2003). A real dilemma in the IT industry is to be able to conduct performance benchmarks to yield accurate hardware requirements at an early stage of development (Seltzer, 1999). For systems that does not yet exist, the only way to quantify the required system capacity to process a given workload is through performance benchmarks; these benchmarks should reflect not only an isolated component, but also a view of the entire system, a benchmark from an application perspective (Darema, 2001).

1.2 Problem definition

The company used as a case scenario in this paper faces the challenge of how to conduct system performance benchmarks effectively. When writing proposals for clients, there are questions related to hardware sizing and performance that need prompt and accurate answers. Unfortunately, there is not an easy way to provide these answers without conducting tests and benchmarks on different platforms and configurations. Benchmarking is a time-consuming and

expensive process, and without the right benchmark metrics and tools, there is a big chance that the results will not be as expected (King, 2005; Null & Lobur, 2003).

1.2.1 Existing performance benchmark process

The following section describes the methods utilized by the company today to conduct performance benchmarks, as well as the different issues related to the present procedures for conducting the benchmarks.

1. The existing way to conduct hardware capacity sizing did not implement a methodology based on current industry practices such as SPEC frameworks.

2. The results from the benchmarks conducted by the company were not as accurate as expected and were only relevant to one specific platform and OS.

3. In order to obtain benchmark results from other platforms, it was necessary to reconfigure all the tools and deploy them on each different platform with different hardware, making the process lengthy and expensive.

4. It was hard to compare results across platforms.

1.2.2 Benchmark tools used

The existing tools to conduct hardware capacity sizing consisted of a number of UNIX shell scripts written using Expect language (The Expect Home Page, 2006). The benchmark analyst used these scripts to monitor the application and count the number of transactions processed per second. The author then had to manually extract the data to calculate the number of transactions from the application audit log.

This benchmark approach presents the following issues:

- The analyst does not have the ability to capture the time it takes for the system to write the data to the audit log.

- The system noise generated by the expect scripts was not considered during throughput calculation.
- The system state was measured taking screen capture of UNIX system tools such as “top” and “perfview” (for more information on top and PerfView see Unix Top [2006] and Using Hp Perfview [2006]).

The table below shows an example of a performance benchmark result conducted by the company.

Table 1 Benchmark Results 8-24-2005

Trans Runtime	Seg/ Inter.	Files In	Total Mesg.	Start Time	End Time	Elapsed Time	Rate /sec	Num/Trans Per hour
8-4	4	200	800	13:57:33	14:00:04	0:02:31	5.30	19,073
8-4	4	200	800	14:05:36	14:07:32	0:01:56	6.90	24,828
8-4	4	200	800	15:51:40	15:53:39	0:01:59	6.72	24,202
8-4	3	200	600	16:07:14	16:09:10	0:01:56	5.17	18,621
8-4	3	200	600	16:12:00	16:14:02	0:02:02	4.92	17,705
8-4	3	200	600	16:19:28	16:21:26	0:01:58	5.08	18,305
8-4	1	200	200	16:29:42	16:31:21	0:01:39	2.02	7,273
8-4	1	200	200	16:33:47	16:35:42	0:01:55	1.74	6,261
8-4	1	200	200	16:40:07	16:41:49	0:01:42	1.96	7,059
8-4	2	200	400	16:45:50	16:47:40	0:01:50	3.64	13,091
8-4	2	200	400	16:48:42	16:50:22	0:01:40	4.00	14,400
8-4	2	200	400	16:54:02	16:55:53	0:01:51	3.60	12,973

The table includes only columns relevant to the benchmark because of space restrictions. The columns organization is as follows:

In the first column, Trans Runtime is the data transformation application runtime and communication bridge runtime. The data transformation is a data format translator very memory intensive and CPU intensive, it loads the document into memory and maps each field to the corresponding target field, and then writes to disk. The communication bridge receives the data after translation and forwards it to its final destination. Figure 1 below represents an overview of the target system components for this benchmark.

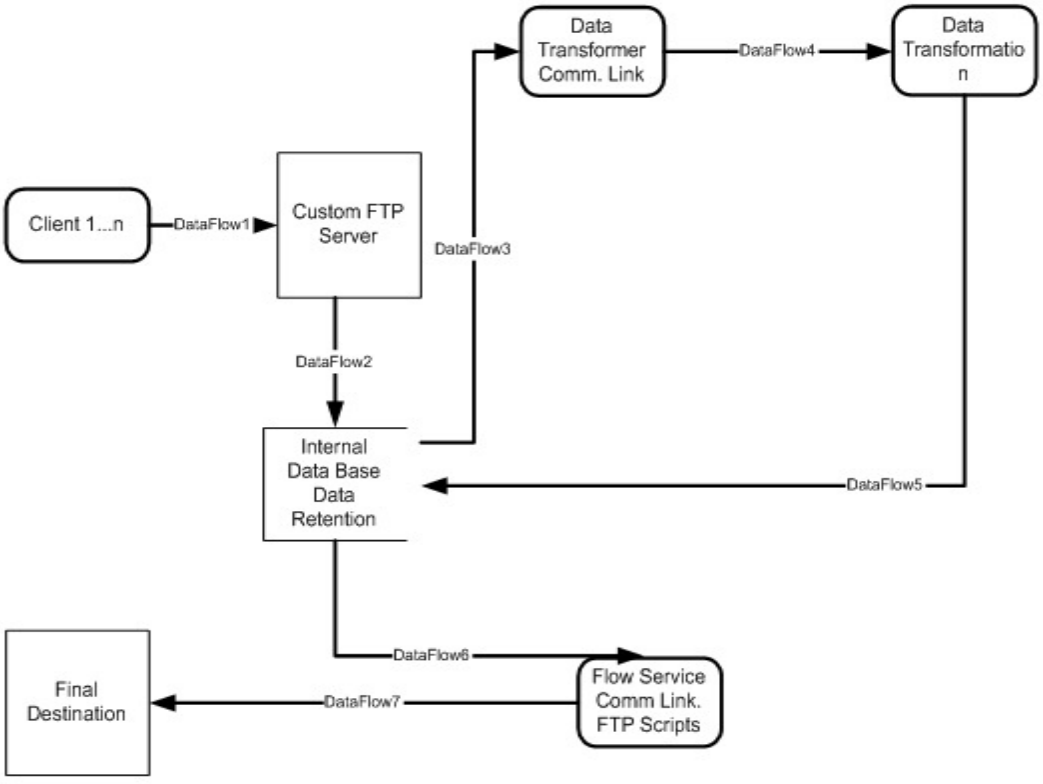

Figure 1: Target system data flow

The Trans Runtime from Table 1 represents the number of concurrencies allowed for the data-transformation communication link and the number of concurrencies allowed for the service-flow communication link. Concurrency refers to the maximum number of simultaneous connections the module will accept.

Column two, Seg/Inter. refers to the workload type, number of segments (transactions) in one interchange. The third column, Files In is the number of files submitted for this particular test. These files are submitted simultaneously using expect scripts. The number of inbound connections to the FTP server varies from test to test in an attempt to simulate a workload distribution somehow similar to the real world.

The total message column represents the number of files times the number of transactions per file. This particular metric is number of transaction per seconds. The next two columns Start Time and End Time are self-explanatory. Starting time is the time when the data arrived at the system, and the ending time is the time the item arrived at the receiver system or final destination. To avoid network overhead during this benchmark, the author used the local system as the final destination as well as a remote system within the same network utilizing a high-speed link.

Elapsed time column shows the ending time minus the starting time, which is equivalent to the time it took to process the workload. The rate/sec column indicates the number of transactions per second for the test, and the last column, Num/Trans is the estimated number of transactions per hour.

1.2.3 Benchmark results considerations

This benchmark does not use a realistic workload distribution; the samples per tests are not enough to represent the real production system, and the hourly estimate does not take into

consideration system degradation over time, if any. It would be very hard and inaccurate to estimate the hardware capacity to process the workload of a production system without considering all the variables involved, such as the following: system state, workload distribution, and system performance degradation over time.

The benchmark does not offer any information on why there was a difference from one sample to another. For example, on the second test, the first sample took 151 seconds, and for the second sample, it took 116 seconds, a significant difference when the metric is transactions per second. Another observation is which result should be take from each test, the higher performance, or the average? Averaging the results could be very misleading since the samples are so small; any large variance would misrepresent the results.

The benchmark does not follow a methodology; it does not consider any benchmarking best practices, and falls short in providing the kind of information needed to make an acceptable recommendation for the necessary hardware capacity to process a production workload.

1.2.4 Effective performance benchmarks

In order to conduct performance benchmarks effectively for any application, it is necessary to understand the characteristics of good performance metrics before adopting any benchmark strategy. The performance analyst must know not only how to measure, but what to measure. Lilja (2000) suggests six characteristics that the author considers very important to observe in a proposed solution:

1. Linearity: Performance is proportional to the increase of the performance metric.

2. Reliability: System A is faster than system B when the performance metric indicates that it should be.

3. Repeatability: The analyst expects the same value of a particular metric through many executions of the same test.

4. Easiness of measurement: The analyst can easily measure the established parameters across platforms.

5. Consistency: The unit of the metric is the same across different systems

6. Independency: The manufacturer of the system because of marketing competition does not influence the performance metric.

The performance benchmark methods utilized by the company today do not employ performance metrics that meet the characteristics suggested above, causing the benchmark results not to be as accurate as expected and increasing the time required to execute the benchmark. The tools utilized are not platform independent as they should be, making it harder for the analysts to perform benchmarks across different platforms. Added to this is the complexity of the existing tools and the cost associated with hardware procurement for benchmarking purposes.

1.2.5 Research goal

This paper demonstrates that it is possible to conduct system performance benchmarks for custom applications effectively if all variables involved are considered and the analyst adopts the right strategy. The author will demonstrate the validity of the statement through the research of existing work on the area of application performance benchmarks, through the analysis of research findings, and the proposal of a methodology that will allow the company to conduct performance benchmarks in order to provide its customers with accurate hardware capacity requirements for its suite of products.

1.3 Business case

Statistics gathered during the latest application benchmarks showed that 40% of the time is spent writing scripts and configuring environments for testing. For different hardware and operating systems, it was difficult to reuse the benchmark tools without modifications to accommodate the new platform; as a result, benchmark outcomes were difficult to relate from platform to platform due to differences in the tools utilized or to the fact that the performed tests were not exactly the same across every platform.

1.4 Business goals

The business goals for this project are to reduce configuration time for testing and benchmark environment, to reduce the number of personnel needed to conduct performance benchmarks, to reduce the technical level of expertise required to conduct testing and benchmarking, and to implement a performance benchmark methodology that provides the necessary tools to accomplish these goals.

1.5 Scope of the project

The scope of the project is limited to researching and analyzing data related to performance benchmark in order to propose a solution to address the issues previously stated. The proposed solution will specifically address methods to conduct performance benchmarks for the company's communication system referred to as System A and the data transformation system referred to as System B. The company will implement the proposed solution on the following platforms: UNIX HPUX11, UNIX HPUX11i, Red Hat Linux AS, and Sun Solaris operating systems. The methodology itself must be generic enough for anyone to implement it on applications running on platforms other than the ones mentioned in this project.

1.6 Outline

The organization of the remaining chapters is as follows. Chapter 2 presents the literature review for this research paper, which includes a performance benchmark definition, the importance of a well-defined set of metrics. Additionally, Chapter 2 describes several computer systems evaluation techniques and the need for performance analysis, the need for performance benchmark from an application perspective, and performance benchmark considerations. Finally, the chapter contrasts and compares existing benchmarks tools such as LINPACK and SPEC CPU among others.

Chapter 3 presents the methodology utilized in this research from a project and a research perspective. The chapter explains the research procedures and methodology and the life cycle method followed during the project. The chapter includes a brief description of the scope of the proposed solution, and explains the results and deliverables. Chapter 3 also depicts the resources available to the project.

Chapter 4 presents the project history, how the author managed the project, financial estimates, and project benefits from a monetary and non-monetary point of view. The chapter analyzes the different changes the project plan went through, the research findings, analysis and the resulting solution. Chapter 5 includes lessons learned, what the author would have done differently in the project, discussion of project expectations, next stage of evolution of the project if continued, conclusions and recommendations.

1.7 Definition of terms

AIM benchmark: AIM Technologies Server Benchmark

BLAS: Basic Linear Algebra Subprograms

CPU: Central Processing Unit, computer main processing chip

DBMS: Database Management System

DP0: Development Phase 0, Starting phase of a development live cycle

FORTRAN: Formula Translation a general-purpose procedural programming language originally developed in the 1950.

HP: Trademark of the Hewlett-Packard Company also known as HP

HP Perfview: Performance monitoring application for HP UNIX operating system

IBM AIX: Brand name of IBM UNIX operating system

IBM DB2: The brand name for IBM database system

I/O: input/output computer subsystem

LAPACK: Linear Algebra Package Software library for numerical computing

MFLOPS: Million floating points per seconds: CPU unit of measurement

MIPS: million instructions per seconds, CPU unit of measurement

NFS: Network File System

OASB: Oracle Application Standard Benchmark: A performance benchmark specific to Oracle Application Server

OLTP: Online Transaction Processing

Oracle Database 10G: Oracle Corporation brand name for their DBMS

Petri net: Is one of several mathematical modeling languages use for representation of discrete distributed systems. Also, know as place/transaction net or P/T net.

PMI: Project Management Institute

Poisson process: A Poisson process, named after the French mathematician Simeon Denis Poisson (1781 - 1840), is a stochastic process, defined in terms of the occurrences of events.

Red Hat Linux AS: Red Hat Linux Advance Server Operating System

SFS: System File Server benchmark

SFS/LADDIS: The SPEC System File Server (SFS) benchmark is a suite of tools specifically designed to measure UNIX system performance running on Network File Systems (NFS) protocol.

SLA: Service Level Agreement is a formal written agreement made between two parties: the service provider and the service recipient. It is a core concept of IT Service Management.

SPEC: The Standard Performance Evaluation Corporation (SPEC) is a nonprofit organization that aims to produce fair, impartial, and meaningful benchmarks for computers. SPEC founded in 1988 and financed by its member organizations, which include all leading computer and software manufacturers.

Stochastic: Stochastic, from the Greek "stochos" or "goal" means of, relating to, or characterized by conjecture and randomness. A stochastic process is one whose behavior is non-deterministic in that the next state of the environment is partially but not fully determined by the previous state of the environment.

Sun Solaris: Brand name for Sun Microsystems UNIX operating system

TCP: The Transmission Control Protocol (TCP) is one of the core protocols of the Internet Protocol suite. *TPPC*: Transaction Processing Performance Council is a not-for-profit organization dedicated to defining transaction processing and databases benchmark.

UNIX HPUX11i: Brand name for Hewlett-Packard UNIX Operating System

(Wikipedia, 2006)

1.8 Chapter summary

Information technology companies involved in real-time data processing applications face the challenge of providing users with hardware capacity requirements at a very early stage of a project, most likely during the requirements gathering phase. In order to provide accurate information about hardware capacity for a system that does not yet exist, it is necessary to base the information on data collected from previous benchmarks if they are available, and try to adjust the data to a particular workload. Unless all variables involved are considered, conducting benchmarks is time consuming, expensive, and prone to misleading results. Variables such as performance metrics and statistical methods used for results analysis are very important.

The purpose of this research is to gather and analyze information related to performance benchmarks, and propose a solution that will allow the company in question to conduct accurate benchmarks in order to provide realistic hardware capacity requirements for its suite of products.

Chapter 2: Review of Literature/Research

2.1 Introduction

System performance benchmarks come in many flavors, many studies and methodologies have been proposed and implemented. However, the process of benchmarking custom applications is still very difficult and time consuming. Difficulties such as misunderstanding of workload requirements for production applications or lack of information in regards to the workload distribution as well as the unavailability of real production data are only a few of the issues faced by system performance analysts.

Even with the myriad of performance benchmark tools and methodologies available today, the controversy continues (Newport, 1995). This research studies existing methodologies for performance benchmarking with the purpose of defining a strategy that will help analysts use an application-oriented benchmarking approach to determine hardware capacity for custom systems.

2.2 Performance benchmark definition

Simply speaking, performance benchmark is concerned with how fast and efficiently a system reacts to a workload, based on a set of metrics established by the benchmark analyst. Lilja considers benchmarking a science and an art due to the extensiveness and the complexity of the field. Lilja (2000) explains in detail the different aspects of system performance benchmarking and acknowledges the importance of measurement, interpretation, and communication as part of the analysis of performance benchmark. Measurement is one of the biggest issues the company faces. It is important to understand not only how to measure, but what to measure when conducting performance benchmarks and it is necessary to have well-defined metrics.

2.3 Performance benchmark metrics

Lilja's (2000) fundamental question on measurement is how to measure the data, using techniques without disturbing the system while maintaining the accuracy and the ability to reproduce the results. According to Lilja, it is necessary to present these results in a clear and consistent way. Lilja presents five tenets of performance benchmark in his work: comparing alternatives, determining the impact of a feature, tuning the system, identifying relative performance, and setting expectations. This research is interested in the last tenet, setting the customer's expectations about the hardware capacity required to run a specific application based on the workload. There are three general techniques to investigating system performance: measuring the system if it exists, simulation, and the analytical model. Lilja compares these three techniques to determine how effective they are in terms of flexibility, believability, accuracy, and cost. See table 2

Table 2 *General performance analysis techniques*

Characteristics	Analytical Model	Simulation	Measurement
Flexibility	High	High	Low
Believability	Low	Medium	High
Accuracy	Low	Medium	High
Cost	Low	Medium	High

From the table above, flexibility in analytical models is high, because it allows for changes without the limitations of changing a production environment, in a simulation environment flexibility is also high, you can change any parameter without affecting production. For measurement, flexibility is low if you are taking measures from the production environment,

there it will be impossible to make changes or to adjust workload, that is why Lilja considers flexibility low in such environment. Believability in analytical models is low, because you are not testing with production loads but with mathematical models, these models will provide estimates that can vary from that of the real production system. In a simulation environment, believability is medium, it is higher than analytical models, but the analyst is not using real workloads and the environment is not exactly the same as the production system.

Accuracy is low for analytical models, as mentioned before, we are dealing with estimates. In simulation models accuracy is medium, because they do not represent the exact production environment. In models based on measurements accuracy is high, if the measurements are taken from the production system or a system similar to production. The cost can be debated, Lilja considers analytical models low in cost, but the author considers this a dependency on the complexity of the system, if the system is very complex, designing accurate models can be an expensive and time-consuming task. The same goes for simulation models, simulating a full system is also an expesive endeavor. For measurements, the cost can be high if different platforms are required, but it can also be less expensive if the production system can be duplicated on a test environment.

Knowing what to measure is paramount to the success of the performance benchmark. Foltier and Michael (2003) group the choices for measuring system performance in two categories: system-oriented and user-oriented measures, where system-oriented is concerned with throughput and system utilization. The user-oriented approach relates to response time and turnaround time. Throughput is the average number of transactions, jobs, or tasks performed per unit of time, while utilization is the measurement of time while the system is busy processing the workload (Foltier et al., 2003). For the purpose of this research, both throughput and utilization

will be of use in determining hardware capacity. Because of the custom nature of the system considered in this research, the user-oriented measurement approach will be also of use. The response time of the overall system from input to output is a key element on a transaction-based system (Foltier et al., 2003).

2.4 Other computer systems evaluation techniques

In contrast with Lilja, Foltier et al. (2003) lists four techniques for computer systems evaluation: analytical modeling, Petri net modeling, simulation modeling and empirical or testbed analysis. Empirical or testbed analysis is equivalent to the measuring approach Lilja refers to. Petri net modeling or place/transition net, also known as PT/net, is a symbolic language used to represent a distributed system in mathematical terms.

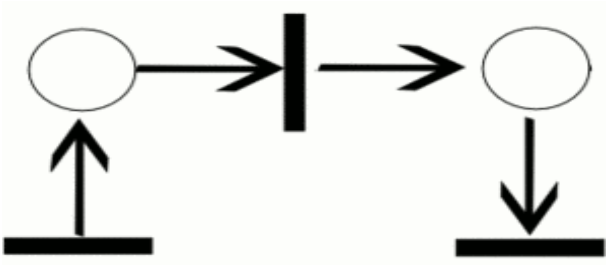

Figure 2: Petri net model example (Petri Net, 2006)

Carl Adam Petri (1962) invented Petri net in his doctorate thesis. Petri net models along with many other mathematical representations for discrete distributed systems are part of the analytical models group. The proposed solution for this project does not implement Petri net models, for more information about Petri net see Petri [1962], and Balbo et al. [2000].

2.5 The need for performance analysis

In general, performance analysis is an ongoing process. In this study, performance analysis tools determine hardware capacity. There are several reasons why system performance

is important. Cassier (2005) defines the need for system performance as a series of questions summarized below.

1. How well is the system running?
2. What is the meaning of running fine?
3. Is the system performing at an acceptable level?
4. Can the system handle future loads?

Another reason to conduct performance analysis has to do with the need to satisfy the business requirements. The business formulates a set of expectations as a contract with the technical department in a Service Level Agreement (SLA; Cassier).

Before the business agrees to SLAs, it is necessary to conduct performance benchmarks and performance analysis to determine the system capabilities. From the author's own experience, it is very important to keep in mind future workload increases when entering in a SLA contract. At the time of writing this paper, the company in question was in discussions to renegotiate a SLA agreement because the preliminary assessment yielded information that after six months in production was no longer valid. The response time for storing a single transaction went from an estimated 800 milliseconds to 1 minute for some transactions during heavy-load peaks, failing the SLA requirements.

2.6 Performance benchmarks from an application perspective

Gibbs et al. (2004) point out some drawbacks of conventional benchmarks including level of abstraction compared to the real production environment and simplification, making benchmarks a good tool for system comparison, but not an accurate tool for capacity planning. According to Gibbs (2004), it is necessary to consider the fact that companies typically deploy

applications on dedicated systems; most likely, the systems have been fine-tuned for maximum performance and for using the latest technologies to improve performance.

2.7 Performance benchmark considerations

The analyst must consider all aspects related to the application performance during the benchmark, and it is not enough to reiterate that the workload must be as close to the real production workload as possible. King (2005) explains how organizations today do not consider or pay too much attention to these performance aspects under the excuse that a little fine-tuning or hardware upgrade will take care of any performance problem. Companies consider benchmark a byproduct of business requirements, technical architecture, development technologies, application design, and the physical implementation of the design (King).

King (2005) agrees with Gibbs (2004) in that a benchmark (viewed from a conventional context) hardly reflects the real production system for which it is intended; they both agree that not enough consideration is given to representing the benchmark environment as a duplicate of the production environment, mostly because of time constraints. King puts it this way when referring to benchmarks: "At best, they may constitute a credible subset of a proposed system whose usefulness is ultimately constrained by a lack of time to build a more lifelike representation; at worst, they can be unrepresentative systems that produce misleading results" (King, 2005, p. 45).

2.7.1 Issues with existing performance benchmarks

For a better understanding of the benchmark methodology ambitioned by the author, it is important to review the three most common types of benchmarks in the industry as cited by King (2000): industry benchmarks, vendor benchmarks, and custom-in-house benchmarks.

The functionalities of industry benchmarks do not equate to the use of real applications, and generally refer to specific parts of a system, like memory, CPU, IO, etc. According to King (2000), they would be useful if they could match the production application, which is difficult if not impossible. Later in this chapter, the author considers some of those industry benchmarks in more detail. The author's opinion is that vendor benchmarks are selling tools, and the main purpose is to show that their product is better than the competition. Software vendors conduct their benchmarks under environments far from the real-world use of the product. King suggests that custom-built and in-house benchmarks have a better chance of producing accurate results, considering the effects of time-constraints and cost.

To avoid some of the problems previously mentioned it is necessary to consider the following items in relation to performance benchmarks (Vokolos, 1998):

1. A workload should be representative of the one the application will use in a production environment. The performance analyst will have to identify where the data will come from, and if this data is truly representative of the real workload.

2. The second issue Vokolos considers is the type of workload the benchmark will reflect, an average workload or very heavy-stress load. He stresses the need to identify the observation window from where the analyst will take the average or stress load. It is necessary to observe the system during peak hours. This observation could be for 12 or a 24 hour period, in some cases it could go for several days in order to obtain a better snapshot of the workload distribution. Keep in mind that some workloads are not present at a particular day or week and sometimes will only appear at month-end.

3. The analyst must consider the environment used for the benchmark. Will it be an isolated environment or a shared environment? This must conform to the future production environment for the application (pp. 80-91).

2.8 Workload considerations

A good representation of the workload the application will process in a production environment is necessary if the benchmark is to produce reliable results. Vokolos (1998) stressed the need for a representative workload and so does Fortier (2003). Fortier uses workloads as the means to test and stress the system and points out the need to develop workloads that faithfully represent the real-world environment in which the system will operate. Additionally he considers workload specifications as the key concern in benchmarking. Foltier (2003) describes workload as the set of all inputs the system receives from its environment. This workload must be capable of stressing the system to its total capacity, and the analyst should be able to sustain the workload for a long period. The analyst must also be able to present the workload to the system in a series of ways that will closely mimic the real system.

2.8.1 Ideal workload for performance benchmarks

Gathering benchmark information from real production systems is almost impossible considering the overhead imposed by data-gathering applications and the fact that we might not see the production system running at full capacity at any given time. Riedl (2001) presents a framework for the modeling, representation, and analysis of real-world workloads. He uses the framework to analyze customer traces on high-performance transaction and communication systems; as a result, these traces show data access patterns that the analyst can use on meta-benchmarks to evaluate real-world applications. According to Riedl, these meta-benchmarks

must specify the procedures to generate synthetic workload traces that will resemble the characteristics of real traces.

This is a promising approach to the issue addressed in this paper. If the benchmark analyst can simulate benchmarks from information gathered on traces of the real application under real workloads, it would be possible to have a better sense on the nature of the workload in a production system. Again, it is good to keep in mind Vokolos (1998) and Frontier's (2003) concerns with the variability of the production workloads. When performing production traces it is necessary to extend these traces for as long as possible to obtain a realistic picture of the overall system workload and keep close observation on the overhead presented by the system trace itself.

Interestingly, Rield's (2001) meta-benchmark result "demonstrates the need for a generic technique that allows us to create the right benchmark for each application scenario" (p. 94).

2.8.2 The need for representative benchmarks

This research shows evidence that there is a need for a universal methodology for performance benchmarks that are representative of the way applications are used in production environments; most of the standard performance benchmarks in use today have specific metrics to measure performance and employ different levels of granularity for given benchmarks, from CPU to IO measurements. These benchmarks do not provide an overall picture of the interaction of system components' and the application or program under different workloads scenarios. Many organizations are integrating granular components into benchmark suites; using these benchmark suites, the analyst could obtain an overall view of the entire system. Dujmović (2001) examines these benchmarks suites and exposes the reasons why they are not as effective as they could be; he proposes that benchmark suites engineers consider the differences between each

benchmark program before incorporating them into a benchmark suite. This gap analysis should include the following subjects:

The White Box difference between benchmark programs which is concerned with the difference between computer workloads presented in the equation

$$d(A,B) = \sum_{i=1}^{N} |U_i^{(A)} - U_i^{(B)}| \; / \; \sum_{i=1}^{N} (U_i^{(A)} + U_i^{(B)}), \quad 0 \le d(A,B) \le 1$$

when abstracting a computer system to a set of N hardware and software resources using various computer workloads where each is characterized by U1…UN, where 0<=1 Ui<= 1, i = 1, N to express the difference between programs A and B, denoted d(A, B), for both non-overlapped and overlapped use of resources Dujmović (2001.)

Dujmović considers the above a white-box because of the "assumed access to all internal components of a computer system relevant to the benchmark, as a consequence, d(A, B) is not the same for all computer architectures and it is difficult to select the most appropriate configuration for measuring U1,….,Un" (Dujmovic, p. 67). The relevance of the above equation to this project is the issues it represents. First, how can a proposed solution mitigate the difference in d(A, B) from system to system, and second, how to implement a workload strategy that will be the same across different computer systems. A good strategy will consequently have to deal with the configuration aspect from system to system. The author proposes a linear or plain configuration across all systems used for the benchmarks.

Dujmovic discusses other differences among benchmark suites such as, the black-box difference between benchmark programs, the size and density of the benchmark suite.

2.9 *Capacity planning considerations*

All the researchers considered in this chapter follow the same goal: to provide methodologies and techniques that will allow users to better plan for hardware capacity for

specific applications. Capacity planning must not only look at the capacity of the hardware, but also its cost. The solution must satisfy both requirements to be effective. Cassier (2005) defines capacity planning as "the process of planning for sufficient computer resources capacity in a cost-effective manner to meet the service needs for all users" (p. 56). A way to meet capacity requirements is to throw more hardware until the system reaches the desire capacity, but how cost effective is this? Additionally, a company might end up with a system that is underutilized.

2.9.1 Questions on capacity planning

Cassier (2005) suggests the following question during capacity planning: How much of your computer resources are used? This is a very important question because it avoids purchasing more resources than needed. If the capacity planning involves the review of an existing system, it is necessary to understand the percentage of resources utilized by the application and not just the overall capacity of the hardware where the system is running. Cassier recommends reviewing CPU utilization, processor storage, I/O subsystem, coupling facilities, channels, and networks.

The next question is, which workloads are consuming your system resources that affect workload distribution? This as well is a very important question because it will identify the pattern of resource consumption and help determine what to do with the specific workload other than increasing the capacity of the system. The system analyst can redistribute, reschedule, or reroute the workload that uses the largest amount of resources to a parallel system (Cassier, 2005). The third question is, what is the expected growth rate? Previously in this chapter, the author provided a real-world example of how important this question is. The company did not accurately consider growth rate during a benchmark and ended up having to renegotiate SLAs shortly after going into production due to a fast increase in workload and slow system response.

The last question is critical considering the company's SLA renegotiation issue: when will the demands on current resources impact service levels? For more on capacity planning see Gibbs et al., (2004), chapter 5 and Cassier (2005).

2.10 Benchmark Programs

This section provides an overview of some of the many benchmark programs in existence today, and their limitations when compared to a methodology that considers the overall application performance on the given system with realistic workloads.

2.10.2 LINPACK

LINPACK is a collection of FORTRAN kernels designed for super computers of the 70s and 80s and used to analyze and solve linear equations and linear least-square problems. LAPACK replaced LINPACK suite in must part due to its ability to run on shared-memory, vector supercomputers (LINPACK, 1970). The performance metrics reported are total execution time and MFLOPS (Lilja, 2000).

2.10.3 Perfect club

A benchmark suite of 13 sets FOLTRAN application programs. Perfect Club derives from computational fluid dynamics, including chemical and physical modeling. Performance Evaluation of Cost-effective Transformation's primary goal was to evaluate compilers (Lilja, 2000). To learn more about Perfect Club see Cvetanovic et al. (1990).

2.10.4 SPEC CPU

Founded by major players in the workstation manufacturing industry, including HP, Digital Equipment Corporation, MIPS, and Sun Microsystems in 1988, the System Evaluation Cooperation (SPEC) is an effort to standardize the way computer manufacturers run and report

benchmark results. To compensate for the rapid advance of computer technologies, SPEC frequently issues a new set of benchmarks. The first set, SPEC89, used four C programs and six FORTRAN programs (Lilja, 2000). SPEC is a very useful benchmark tool for specific computer components like CPU, graphics, or applications like Java client/server, mail servers, network file systems and Web servers, but it is not possible to tailor it to support custom applications and custom workloads. This benchmark is far more reliable than vendor benchmarks because it comes from an independent organization. However, it is important to notice the fact that Lilja mentions: computer manufacturers are familiar with these benchmarks and will fine-tune their products to respond well to those benchmark specifically, resulting in a false impression that the system will perform with the same capability when running a production application.

2.10.5 TPC-C

A benchmark that relates to the problem discussed in this paper is the Transaction Processing Council (TPC). The TPC-C benchmark is an OLTP or Online Transaction Processing benchmark. The benchmark aims database systems where users submit queries to update or retrieve information, such as automatic tellers, inventory-control systems, etc. It reports throughput requirements, with a metric of number of transactions per minutes. This research targets a similar system where the system processes transactions submitted via a number of communication means; the transactions are processed and the output forwarded to a back-end application or a remote system. TPC-C supersedes TPC-A; it has been enhanced to keep up with later transaction processing technologies.

The TPC-C benchmark goal is to provide a set of functional requirements that the analyst could implement regardless of the hardware or operating system. The issue faced with TPC-C, as a potential solution to the problem discussed in this paper is that TPC-C specifically targets a

transaction-processing database and is not necessarily customizable for a custom product as the one considered here. TPC-C provides performance benchmarks for transaction processing systems running Oracle Database 10G Enterprise, IBM DB2 UDB 8.1, Microsoft SQL Server 2000 on different platforms such as IBM AIX, Red Hat Enterprise Linux 4 AS, Microsoft Windows Server 2003, HP UX 11.iv2 64-Bit Base OS. The metric reported by the benchmark is transaction cost. For more information on TPC-C and all other TPC benchmarks.

2.10.6 SFS/LADDIS

SFS/LADDIS a client/server benchmark used to generate commands and sends them to a server running on an NFS file system (Lilja, 2000). SFS97 is the latest benchmark, for more information about this specialized benchmark see Spec Sfs Suite (1995).

Lilja (2000) provides detail about the following other popular benchmark programs in use today: AIM Benchmarks for UNIX servers tests, Business Benchmark for UNIX server's throughput, WinBench is a benchmark for Windows PC graphics, WinStone, a benchmark for Windows PC performance

2.10.7 Existing benchmark tools issues

The main issues with the tools presented here are that they employ predetermined applications with predetermined workloads. They are useful benchmarks to compare one computer system to another, but they are not useful as tools to provide information about the hardware capacity needed to run a specific application.

2.10.8 Vendor specific benchmark tools

Some vendor specific benchmarks in the market today are SAP Standard Application benchmark, used to test scalability of mySAP Business suite; Oracle Application Standard benchmark (OASB), which demonstrates the performance and scalability of Oracle Applications,

and Siebel platform sizing and performance program benchmark. The issue with these benchmarks is that in many cases the vendor influences them in order to compete with other vendors (Gibbs et al., 2004).

2.11 Project contribution to the field of information technologies

The purpose of this research is to analyze the various benchmark strategies and the work done by other researchers in the field of Information Technology to derive a methodology that will help the company and other IT professionals conduct performance benchmarks that more closely reflect how an application will perform in a production environment. Such methodology will help not only the company used in the research, but others looking for a methodology that could adapt to the needs of those analysts executing benchmarks to determine system capacity, especially, systems running custom applications.

2.12 Chapter summary

The review of existing benchmark methodologies in this chapter shows the difficulties in conducting benchmarks representative of real-world production environment. Industry benchmarks are a very good way to compare hardware, but they do not necessarily represent how a particular application will perform when utilizing real-world workloads. These benchmark tools test specific parts of a system such as memory, graphic cards, CPU, etc.

Vendor benchmarks are targeted to prove that a particular vendor application performs better than another does. In other words, they are used as selling tools. This chapter also demonstrates that sufficient research has been conducted to provide solid strategies, that if combined could form the framework for an effective benchmark implementation. In subsequent chapters, the author examined the finding from the research in order to explore a better approach as the one utilized today by the company.

Chapter 3: Methodology

3.1 Research methods

This research adopts McKenzie's (2000) research cycle model from his book *Beyond Technology*. McKenzie's model includes the following processes: questioning, planning, gathering, sorting and sifting, evaluating and reporting.

Questioning: During the questioning phase, the bases for the research were established and the author formulated a research question or thesis statement.

Sorting and Sifting: During this phase, the author gathered and sorted the data by its relevance to the problem.

Synthesizing: The process of arranging the pieces of information found during the research into meaningful information patterns that provide a general view of the issue.

Evaluation: During the evaluation phase, the researcher analyzed the data resulting from the observation of the system in question and reviewed the results of benchmarks previously conducted with the purpose of comparing the data to the findings from the literature research, evaluating the pros and cons.

Reporting: The reporting phase included the drafting and presentation of the solution, including any further analysis for future enhancements.

3.1.1 Specific procedures

Case study

The author uses the company and its systems as a case study and as the scenario to conduct any needed empirical tests.

Grounded-theory

This research utilized grounded-theory methods to support its conclusions. Grounded theory looks to "ground" the theory on data gathered and analyzed, in order to support the theoretical assumptions made in the thesis statement. The author collected data from journals, articles, books, and papers related to the subject matter, as well as documented system tests, observations, and empirical approaches.

3.2 Life-cycle models

The research of the problem resulted in the proposal of a methodology for the company to conduct benchmarks. For the development of system interfaces and benchmark tools needed to implement the methodology, a prototype-oriented Software Development Life Cycle was used. This SDLC uses the following phases:

1. Requirement gathering
2. Requirement analysis
3. Prototype design
4. Prototype development
5. Prototype evaluation

The prototype development and evaluation goes through a 3-iteration process, DP0 through DP2; at the end of DP2, the application must be ready for production deployment. Figure 2 on next page details the proposed life cycle model.

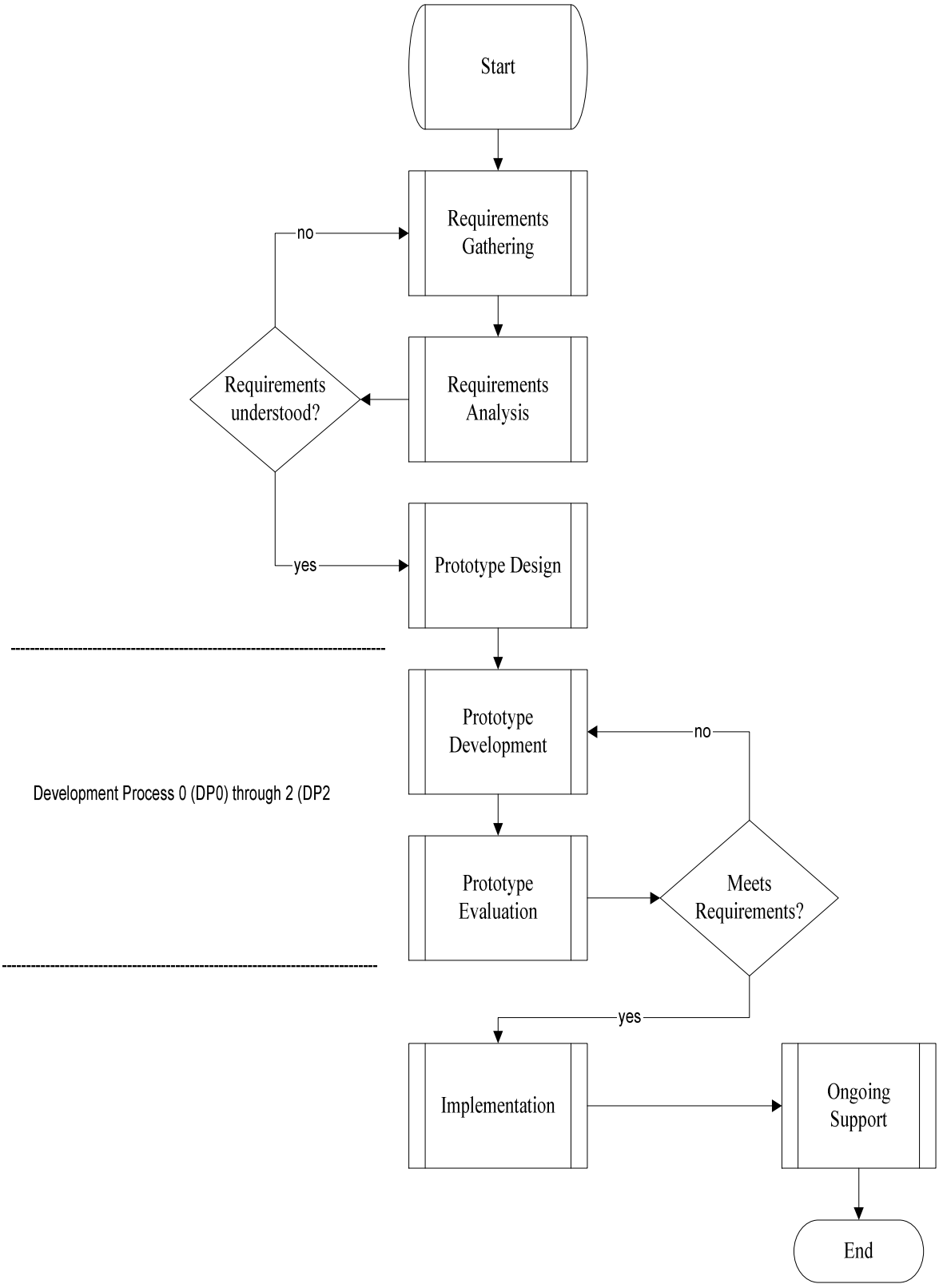

Figure 3 Application development life cycle

3.3 Results/ deliverables

The deliverables from this project include a methodology for hardware capacity planning in the context of custom applications. The proposed methodology includes requirements to conduct benchmarks and a high-level solution design indicating how the methodology could be implemented in the company's existing environment (see Table 2).

Table 3 List of Deliverables

Deliverable Number	Deliverable Name	Comments
1	Research findings	
2	Data analysis	
3	Proposed Solution	
4	Proposed Development Plan	

The research findings lay how other institutions addressed the specific problem or any study relevant to the issue, tools available today to help solve the problem, or the need to develop new tools. Based on analysis of the research findings, the author gathered the positive characteristics of existing methodologies, and proposed a strategy to address the benchmark issue faced by the company. The author also proposed a development plan to implement the solution.

3.4 Deliverables from software development life cycle

The following sections describe the SDLC deliverables for each phase.

- Requirement gathering: The goal of this phase was to understand the problem and gather all necessary requirements. The deliverable was the formulation of the Application Requirements Document.

- Requirement analysis: Understand and prioritize each requirement in order to determine scope. The deliverable for this phase was the final draft of the Application Requirement Document.
- Prototype Design: Complete prototype design making sure it includes all mutually agreed requirements. The deliverable for this phase is the Application Design Document.
- Prototype Development DP0 through DP2: Complete code, unit testing and integration. The deliverable for these phases is a functioning application prototype.
- Prototype Evaluation DP0 through DP2: Validate prototype according to specifications. The deliverables are a prototype that meets all the requirements and is ready for implementation, or a list of requirements that the prototype does not meet.
- Implementation: Prototype implemented on production environment. The deliverable for this phase is a full production application.

3.5 Resources

The project had very limited resources, even when it served a good purpose for the company; the intent was to use existing resources to do the research as well as to develop any needed software. Table 4 briefly describes the resources used for the project phase of this research, including hardware, software, and human resources.

Table 4 Resources

Resource Type	Resource Description	Comments
Hardware	HPUX11 Class A Server	
Hardware	HPUX11i Class A Server	
Hardware	HP Blade	
Hardware	IBM ThinkPad Notebook	

Hardware	Sun Solaris Ultra 10 Server	
Software	Red Hat Linux 3.x Server	
Software	HPUX11 UNIX OS	
Software	HPUX11i UNIX OS	
Software	Microsoft Words	
Software	Custom applications A and B	
Personnel	The author	

In order to publish this work, the author did not include the name of the company or of any application or system used in this paper; for the purpose of this research, the author refers to the organization as "the company". The two custom applications considered the core of the system are application A and application B. Any other component within the data interchange gateway is referred to as component 1, component 2 and so on. For more information, see System Overview in Chapter 4.

3.6 Outcomes

There were several outcomes expected from this research, first a methodology or framework to help the company conduct benchmarks accurately, second a software design to implement the methodology and third, a development project to create the necessary tools to conduct benchmarks following the proposed methodology.

3.7 Chapter summary

This chapter described the research methods used to investigate the issue addressed in this paper. The research utilized a combination of case study, literature review, and grounded theory to gather and analyze the findings. The research followed a life-cycle model based on

McKenzie's (2000) that includes the following phases: questioning, planning, gathering, sorting and sifting, evaluating, proposing and reporting.

As result of this research, a project was executed with the purpose of developing tools to implement the suggestions made by the author. The development phase used a prototyping model based on seven phases or steps: requirement gathering, requirement analysis, prototype design, and development, prototype evaluation, implementation, and ongoing support.

Chapter 4: Project History

4.1 How the project began

This project began in May 2005, after the author conducted several system performance benchmarks for several of the company's customers. The benchmarking exercises lasted for more than two months; the results were relatively accurate for specific platforms and specific hardware. In order to conduct the benchmark and due to the inability to conduct benchmarks on the company's premises for lack of proper equipment, the company rented time from Hewlett Packard laboratories in Detroit, Michigan.

After the fact, other customers requested several performance benchmarks for different platforms. It was difficult to obtain comparable results on different hardware and operating systems, and it turned out to be very expensive to rent laboratory space every time the company needed to conduct a benchmark. As a result, the author of this paper suggested the need to conduct a research in order to find the options available to the company to conduct its own benchmarks in a flexible enough environment that would allow for timely responses and accurate results. The research made it clear that many variables needed to be considered during the benchmark. In addition, that others have performed a large amount of work in the industry but the existing benchmark methodologies did not offer the level of flexibility necessary to accomplish the required results. The author envisioned a methodology that could combine the positive characteristics of existing performance benchmark methods and the development of a set of tools to implement the methodology.

4.2 Project management

As stated in previous chapters after the research finished, and the analysis of the data ended, a project started to develop tools that implemented the proposed methodology resulting

from the research. This project was managed following PMI ("A Guide to the," 2004) guidelines for project management. The project included the following knowledge areas from the PMI guidelines: integration management, scope management, time management, and risk management. Within each of these phases, the project implemented the following PMI processes: initiation, planning, execution, monitoring, controlling, and project closing. The following section briefly describes the project management plan, schedule, and milestones.

4.2.1 Project milestones

Table 5-Estimated Project Schedule (Milestones)

Event	Estimated Starting Date	Estimated Duration
Project Proposal Approved	05/10/05	8 weeks
Requirements Gathering	05/30/05	1 weeks
Requirements Analysis	6/20/05	1 weeks
Prototype Design	7/11/05	1 weeks
Prototype Development (DP0)	8/11/05	2 weeks
Prototype Evaluation DP0	10/11/05	1 weeks
Prototype Development DP1	10/25/05	2 weeks
Prototype Evaluation DP1	11/07/05	2 weeks
Prototype Development (DP2)	11/21/05	2 weeks
Prototype Evaluation (DP2)	12/05/05	2 weeks
Product Implementation	12/19/05	1 week
Project close-out	12/26/05	1 week
Project ongoing Support	January 2006	
Total Project Duration		**24 weeks (6 months)**

4.3 Financial estimates

The return on investment for this project is very positive, as stated in the project overview, as no funding existed for this project. The company used existing resources including hardware and software. The company used existing licenses for third-party applications needed for the project, and the author as a requirement for graduate program completion provided the work force required at no cost to the company.

4.3.1 Monetary benefits

The quantification of the monetary benefits for this project is as follows: After full implementation of the application, the number of hours required to complete a given benchmark would be reduced 30%, or an estimated 96 hours. Estimating an analyst salary of $50.00 per hour, the savings amounts to $4800 dollars per benchmark, if the company conducts 10 product benchmarks per year, the accumulated savings will be $48,000 dollars.

After completing product benchmarks for all supported platforms, the data serves as reference for future configurations providing savings equivalent to an estimate of 60 %t of the 320 man/hours utilized to conduct a benchmark or $96,000 dollars.

4.3.2 Non-monetary benefits

The non-monetary benefits include but are not limited to: Ability to easily conduct performance benchmarks on new hardware or platform, a repository of benchmarks results available company wide, including product benchmark information for all supported platforms, consistency, portability and flexibility.

4.4 Changes to project plan

Originally, the author conceived this project as a full software development project to address the hardware capacity planning for custom applications within the company. Later the

author considered that research was required to gather information about industry methodologies and standards already in place in order to accelerate the outcomes of the project. After researching the different options, it was determined that neither applications nor tools could fully satisfy the present need. As previously observed in the literature review, many benchmark tools exist on the market today, but they have been specifically designed to measure pieces of the system, such as CPU, IO, graphics, or specific applications.

The need for a benchmark tool that would measure custom metrics for the overall application, gather the data, and use it to determine hardware needs for future workloads, was not going to be satisfied by the existing tools. Unless the researcher considered all the variables involved in the hardware capacity planning process, he development of benchmark tools would not satisfy the requirements. The new project plan deliverables include a methodology to conduct performance benchmarks to provide hardware capacity sizing for custom applications.

4.5 Evaluation of project goals

This section evaluates whether or not the project met the expected goals. Before starting the evaluation, it is a good idea to summarize here the expected research goals as stated in Chapter three.

- A methodology or framework to help the company conduct performance benchmarks for custom applications.
- Develop a software design that implements the proposed framework.
- A development project that would build the necessary tools to conduct the benchmarks, these tools would follow the proposed design and methodology.

4.5.1 The proposed methodology

The project accomplished the goal of providing a framework or methodology the company can follow when conducting system benchmarks to provide hardware capacity planning for its customers. Below the author details the proposed methodology.

This research did not intend to develop a new performance benchmark methodology from scratch, but to analyze existing benchmarks, borrow their best methods and combine them into a framework that provides the analyst with a flexible and adaptable approach.

The proposed methodology will address the following topics: benchmark scope, metric definitions, data gathering, workload considerations, data analysis, and results reporting. Considering the issues found in Chapter two with existing benchmark tools the author proposes a methodology based on an application context. An application context means that the main tool utilized to gather the desire metrics will be the production application itself. This methodology will not utilize benchmark kernels or application snips, a full production application will be the base of the benchmark.

4.5.2 Why benchmark from application perspective?

Remember Gibbs concerns in Chapter two where he points out the issue with the level of abstraction found in conventional benchmarks when compared to real production environments (Gibbs 2004). King states that conventional benchmarks hardly represent the real-world use of the applications they target (King 2005). King also emphasizes the difficulties to equate industry benchmarks functionalities to the use of real production environment, and the fact that these benchmarks generally refer to specific parts of a system, like memory, CPU, IO, etc. Because of

these reasons and others detailed in Chapter two, this research considers using an application-based approach.

4.5.3 Benchmark scope

Application oriented is the central theme of this methodology; the implementation includes an application perspective as suggested by Darema (2001). The analyst will use the same application implemented in the production environment as the main tool to process the workload, contrary to kernel benchmarks. The scope of the benchmark also covers the state of the operating as explained by Newport (2005), including CPU and IO utilization, and memory.

4.5.4 Methodology metrics definition

This section addresses the "What to measure?" question. The metric definition falls within the scope of system-oriented and user-oriented metrics as explained by Foltier et al. (2003). This methodology considers both system-orientation to obtain information about system utilization and throughput (transactions/unit of time) and user-orientation to consider turnaround time for transactions requiring acknowledgments.

A reason to consider both types of metrics is the custom nature of the application considered in this research. The target application is a transactional system and the main purpose of the benchmark is hardware capacity planning. This requires not only throughput information, but also system utilization and turnaround time.

The proposed methodology is metric-oriented in nature; the analyst will use an environment similar to the production environment and will gather data about the metrics covered in this section. From this point of view, the methodology falls into the measurement category of Lilja's (2000) General Performance Analysis Techniques. In this category, flexibility is low, believability, accuracy and cost are high. After integration of other components of the

system such as the benchmark data analyzer, the measurement characteristic will take advantage of the simulation characteristics of Lilja's techniques, reducing the cost and increasing flexibility.

The research considers the following metrics within system utilization including CPU, memory, IO, and network interfaces performance. The throughput metric measured the number of transactions per seconds for specific workload size and complexity. Workload complexity is important due to the data transformation part of the application; depending on the data characteristics, data transformation may require longer to process. Response time is the time it takes for a transaction response to reach its final destination.

4.5.5 Methodology data gathering approach

To collect benchmark data, the methodology adopts a tracing benchmark approach as suggested in Riedl (2001). Trace benchmarks use application code modifications to obtain the benchmark data in a way that minimizes the unwanted noise caused by benchmark tools on the target system. The base application requires modifications to generate traces for each resource consumer. The analyst should also consider how much noise trace enabled components generate. The tracing code must generate traces at predetermine intervals to minimize system resource consumption. Trace collection must expand through a significant period to cover all possible workload scenarios on the target system. If the information about the workload patterns does not exist, then the analyst should include as many variations as possible, including average, and very high stress system loads as indicated in Vokolos (1998).

It is extremely important to conduct the benchmark using sample data identical to the data expected in production. In many cases, the expected system response is different from that of the benchmark due to the use of very simplistic test data.

4.5.6 Methodology workload considerations

The methodology also includes the study of the workload dynamics and characteristics in order to establish a workload representative of the real-world environment as suggested in (Lilja, 2000).

The analyst must consider the following when selecting the workload for the benchmark:

1. Workload type: Consider any data type including single transaction per file and multiple transactions per file.

2. File size: A base file size and a measure of sizes representative of the production workload.

3. Data type and complexity: Classify the data type based on the data transformation complexity. For the purpose of this research, the number of fields needing manipulation during data transformation will determine complexity. Simple transaction includes 1 to 100 fields; an intermediate transaction contains from 101 to 999 and anything greater than 1000 is a complex transaction. Of course, the analyst can adjust this matrix if more accurate information exits.

4. Workload distribution: The benchmark analyst must consider the workload distribution; the author recommends a Poisson stochastic model to workload distribution due to its independent event properties and the use of exponential distribution to represent event arrivals (Foltier & Michel, 2003).

4.5.7 Methodology data analysis strategy

Avoid apples to oranges comparison during data analysis. Observe and consider systematic as well as random errors (Lilja, 2000). The author suggests the proper use of statistical formulas such as harmonic means, geometric means, and arithmetic means; proper use

of statistical formulas means to consider when it is appropriate to use them as explained by Lilja (2000).

Use geometric means when summarizing measurements with a wide range of values, arithmetic means when summarizing execution time, and harmonic means when summarizing rates. Notice any system tuning before and after the benchmark and consider any variances due to system performance changes generated by data collection tools.

4.5.8 Results reporting method

Present benchmark results in a clear and simple manner (Foltier & Michel, 2003) avoiding ambiguity. The analyst must search for answers explaining any significant variations in results, it is important to present all aspects of the benchmark including environment and metrics to avoid confusion and misinterpretation of results.

4.5.9 Proposed system design

This section briefly describes the accomplished system design to implement the proposed methodology. The project design document contains detailed information.

After the formulation of the proposed methodology, the author proposed the system design shown in the following graph as the blue print to follow during the development project that would build the necessary tools for the benchmark suite.

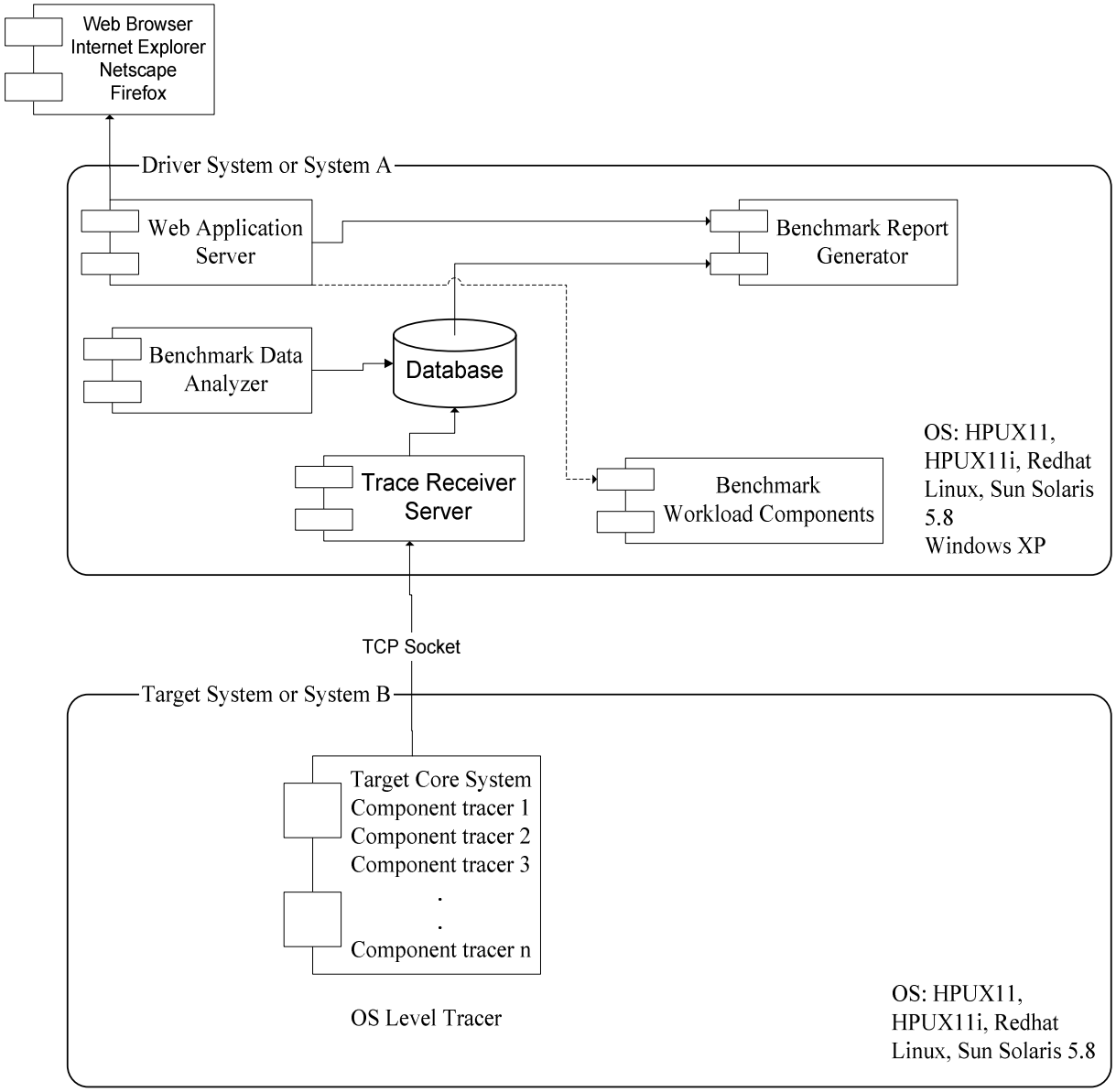

Figure 5: Proposed application design

To understand the above diagram the author provides a description for each component in the next section.

4.5.10 System A: Driver

1. Web Application Server: Will encapsulate all the classes required to interface with the system as well as the benchmark report generator components.

2. Benchmark Report Generator component: A web server service that will extract information from the database and prepare a report for a particular benchmark instance.

3. Database: The tracer server saves all benchmark traces in a database for further analysis and generation of reports.

4. Benchmark Workload: The users will access the benchmark workload component through the web interface for configuration purposes. The workload generator interacts directly with the target system simulating a custom workload to the different target system components.

5. Trace Receiver Server: A TCP socket server that will read incoming traces from the tracer components and write the trace information to the database.

6. Benchmark Data Analyzer: The analyzer is the most complex component of the system. It will read the trace information from the database and calculate the benchmark results. The analyzer will also serve as a model to estimate hardware requirements for different workloads without the need to perform the actual benchmark. The Benchmark Analyzer will be also responsible for generating the hardware capacity recommendations based on the performance benchmark results for a given workload.

4.5.11 System B: Target System

1. Target Core System: This is the application software to be benchmarked.

2. Component tracer 1, 2, 3 … n: Represent the different application components that process the workload and consume system resources.

3. OS level tracer: A system monitor that will report how the system is performing at the OS level using metrics such as CPU, memory and IO usage.

4.6 Development project accomplishment

The goal of this project was to develop a suite of tools to implement the proposed methodology; the suite of tools followed the proposed design. At the time of writing this paper, the project advanced very little toward its ultimate goal. From the proposed design, the following components were required:

System A: The driver system requires a web application server, database, benchmark load components, trace receiver server, and the benchmark data analyzer. Of all these components, only the benchmark loader was developed.

System B: The target system requires the implementation of tracing ability for all application components involved in data transfer and data manipulation, as well as a trace enabled operating system monitor. Out of these components, only one has the required tracing capability.

The author conducted very primitive benchmarks with the components implemented so far. Unfortunately, not enough data exists to quantify how effective the proposed solution may be until all traces are enabled for the application components.

4.7 What went right and wrong in the project

Overall, this project went relatively smooth, without any major issues. The research phase was very effective, and the resulting methodology even when not fully tested, promises a very good contribution to the company and the field. Unfortunately, due the author's engagements in other projects with a higher priority, not all the required modules were implemented, but the company now has a methodology to follow when conducting benchmarks,

either with the proposed design or through any other means. The framework will serve as a guide to effective benchmarks.

4.8 Project variables and their Impact

The variable with the major impact in this project was time. The availability of the author to conduct the necessary development tasks was very limited. The author was the only resource available to this project.

4.9 Findings, analysis and results

During the project, it was obvious that the author was going to face problems due to the lack of human resources to conduct some of the project tasks. It was evident that the project did not have the priority as other customer's related projects in the pipeline at the time. The positive aspect of these findings is that performance benchmarking is an ongoing endeavor. The company is still receiving requests for performance benchmarks and there is a very high possibility of allocating more resources to continue the implementation of the proposed methodology and design.

The project produced a detailed project plan, requirements document and a complete software design document. The author developed two components of the system as shown in section 4.7, if the project is to continue, the developers will integrate these modules to the rest of the system for a full benchmark suite.

4.10 Summary of results

The author summarizes the results from this project in this section as follows:

- The research produced a detailed methodology framework to conduct performance benchmark for custom applications.
- The author presented a detailed software development design for a suite of benchmarks tools to implement the proposed methodology.
- Due to lack of resources to continue development, the effort was suspended after the creation of two of the required components.

Chapter 5: Lessons Learned and Next Evolution of the Project

5.1 What you learned from the project experience

The main lesson learned through this project has been the understanding of the complexity involved in the analysis of system performance. It is not an easy task and requires multidisciplinary knowledge, including knowledge of the system structure, hardware, software, and business requirements. It is also necessary to have sufficient mathematical and statistical knowledge in order to analyze the benchmark results. From a project perspective, the author learned that it is necessary to keep your eyes on the scope of the project and plan accordingly. For a project this size, it was probably a good idea to get more people involved in order to accomplish all phases of the project.

Another interesting fact learned during this exercise was that if accurate and unbiased performance benchmark is required, a better approach is to conduct in-house benchmark following a methodology as the one proposed in this paper including an environment similar to the production system and a workload representative of the real world workload expected for the system.

5.2 What would you have done differently in the project

The allocation of more resources is the most important part affecting the project. It would have been a good idea to get buy-in from managers by selling the project to them in a way that they can see it not only as a benefit for the benchmark analysts, but also for the entire organization and get all the resources necessary to complete the project. Another important aspect is that the project needed to start at the research phase, instead of jumping into a development phase without fully understanding the problem; this caused the project to fall behind from the beginning.

5.3 Initial project expectations

The research met the goal of formulating a methodology framework to help the company perform benchmarks. The project met the expectation of generating detailed software design but fell short to fulfill the expected goal of developing a full suite of tools that implement the proposed methodology.

5.4 Next evolution of the project

The author does not recommend a new evolution phase for the project until all development efforts are completed and the proposed methodology is proven empirically effective. What the author proposes is a sequence of events on how the development effort should continue. Implementation of all tracer components first, followed by the trace receiver server and the database, next the web interface should be implemented, and last, the benchmark data analyzer, which will act as a simulation module using the data from old benchmarks to formulate recommendations on hardware capacity, based on workload information.

5.5 Conclusion

Performance benchmarking as well as hardware capacity planning are very complex endeavors and can be subject to bias and subjectivity. Commercial benchmarks mostly reflect the performance superiority of a system over the competition. Lacking the characteristics of real-world workloads, industry independent benchmarks do not represent a true production environment. A way to obtain accurate, unbiased results is to conduct in-house performance benchmarks. However, how the benchmark is conducted is the most important aspect of the endeavor. This section summarizes what should be done in order to conduct a performance benchmark representative of what goes on in a real-world production environment.

5.5.1 A comprehensive methodology

The first thing is to know how to conduct the benchmark. To follow a benchmark methodology as the one proposed in this paper is the best way to gain knowledge about benchmarking effectively. It is not enough to run scripts to gather data if the data is not going to be analyzed properly. For a methodology to be effective, it must have the following characteristics:

The methodology must enforce benchmarking using an application perspective as explained in this paper. It must employ a good metric definition strategy. It must have an effective data gathering approach. Workload consideration should include workload type, complexity, and distribution, the data analysis must follow a rigorous framework, and the analyst must ensure that the result reporting is clear and unambiguous. The methodology proposed in this paper contains all these characteristics making it a very promising framework for performance benchmark for custom applications.

5.5.2 The right benchmark tools

The benchmark analyst must have at his or her disposal all the necessary tools to conduct the benchmark. It is necessary to count with a suite of tools that observe the characteristics of the proposed methodology, especially for the workload distribution, the data gathering, and the data analysis. This paper proposes a design that a benchmark analyst can use to develop the tools needed to conduct the benchmarks.

5.5.3 Time allocation

Time is a very important factor when conducting benchmark. The analyst must allocate as much time as possible to be able to observe the workload distribution and to pay attention to how the system reacts. It is not enough to run the benchmark for one or two hours or at intervals

imposed by the analyst. The interval, at which the benchmark runs, is dictated by the workload distribution and not by the benchmark analyst. Allocate enough time to analyze the data and to understand the reason for any changes in results as recommended in the methodology proposed in this paper.

5.6 Recommendations

The author recommends that the development phase continue following the sequence recommended in 5.4 and that the methodology is tested empirically as soon as possible to confirm how effective it is.

5.7 Chapter 5 summary

This chapter reviewed the lesson learned through the project, what went right and wrong , what could have been done differenrently and also discussed how important it is to have enough resources in order to accomplish the project goals. The author commented on the importance of carefully observing the project scope. The chapter also includes the need to follow a well-defined framework or methodology during the execution of performance benchmark. The author discussed the initial project expectations and the evolution of the project, and concludes the chapter with recommendations for analysts conducting benchmark for custom applications. In the conclusion, the author recommends strategies such as using a comprehensive methodology, using the right tools, and allowing enough time to conduct the benchmarks.

Appendix A

Annotated Bibliography

Aho, M. & Vinckier, C. (1998). Computer system performance analysis and benchmarking.

This study compares the performance of several disk drives using benchmarks. It explains the use of benchmarking tool for hardware performance and introduces benchmarking tools such as Winstone 97.

Cassier, P. (2005). *Effective zseries performance monitoring using resource measurement facility*. Armonk, NY: IBM Red Books

The book describes the use of RMF and provides case scenarios where RMF is used as a data-gathering tool for IBM and Linux platforms. This is a good example of data gathering applications in use today, their effectiveness, and their drawbacks. This application was used on this thesis as an experimental component to determine the effect of data gathering applications on performance benchmark results.

Eigenmann, R. (Ed.). (2001*). Performance Evaluation and Benchmarking with Realistic Applications*. Boston: The MIT Press.

A compilation of the most recent research on an underlying theme of computer performance and design of management benchmarking. Discusses performance evaluation and benchmarking methodologies describes the need for realistic application benchmarks.

Eigenn,R., Gaertner,G., Saied,F., & Straka, M. (2001). *SPEC HPG benchmarks: Performance evaluation with large-scale science and engineering applications.* Boston: The MIT Press.

This paper discusses the need for benchmarks from a whole application perspective; it explains the differences between kernel, algorithm and small application benchmarks and those in the context of a large computational problem. It describes the contributions of the SPEC High Performance Group through benchmark frameworks such as Perfect Benchmark, Parkbenchmark, SPEChpc and others.

Foltier, P. J. & Michel, H. E. (2003). *Computer system performance evaluation and prediction*. Burlington, MA: Digital Press.

This book describes various approaches to performance analysis methodologies applied to various stages of computer system design. The book presents tools and techniques to be applied to the computer system's life cycle in order to analyze alternatives to optimal performance solutions.

Gustafson, J. & Todi, R. (2001). *Conventional benchmarks as a sample of the performance spectrum.* Boston: The MIT Press.

A study of the major system performance benchmarks existing today. The paper compares and contrasts different benchmarks, their purpose and contributions to performance evaluation, among the benchmarks included in this study were HINT, SPEC, Linkpack, NAS Parallel Benchmark, Peak FLOOPS, STREAM, LLL, Whetstone, Fhourstones. The study is considered a benchmark of benchmarks and depicts the pros and cons of the benchmarks sited above and the correlations among them.

Justos, G. R., Delaitre, T., Zemrely, M. J., & Winter, S. C. (1999). *Accurate performance prediction using visual prototypes,* London: Concurrency.

This paper describes the use of performance modeling for parallel and distributed systems from a prototyping approach. It explains the advantages of performance modeling using visualization prototypes to simulate real world environments, as well as the reasons why to use a prototyping approach to performance modeling. The paper implements the Environment for the Design and Performance Evaluation of Portable Parallel Software (EDPEPPS) based on graphical design, simulation, and visualization to put forward a new model based on extended graphical and textual description.

Kozminski, K., Duewer, B., Lavana, H., Khetawat, A., & Brglez, F. *REUBEN: A Tcl-based reusable environment driven by benchmark applications CAD benchmarking laboratory (CBL*). Raleigh, NC: Department of Electrical & Computer Engineering, NCSU.

This paper introduces an environment for encapsulation and benchmarking of prototype algorithms in a context of realistic design flows. Application-specific parsers read the benchmark description; standardized report generators summarized the benchmarking experiments. The environment described in this paper addresses three basis tenets of benchmarking: Unambiguous definitions of the benchmarking objectives, unbiased benchmark descriptions and a well defined, easy to implement benchmark methodology.

Kumar, S., Pires, L., Ponnuswamy, S., Nanavati, C., J. Gousky, Vojta, M. et al.(n.d.) *A benchmark suite for evaluating configurable computing systems status, reflections, and future direction.* Minneapolis: Honey Well Technological Center.

This paper presents a benchmark suite for evaluating a configurable computing systems' infrastructure, both tools and architecture. The paper explains the use of stressmarks, and benchmarks focus on specific characteristics or property of interest.

Lilja, D. J. (2000). *Measuring computer performance: A practitioner's guide*. Minneapolis: Cambridge University Press.

This paper describes the goals of measuring computer performance and details solution techniques to ensure good performance metrics. The paper discusses means versus metrics in performance measures; it also covers typical errors in performance metrics such as accuracy, precision and resolution as well as sources of errors, quantifying errors, etc. Other subjects covered in the paper are type of benchmark programs, benchmark strategies, regression models and design of experiments among others.

Newport, J. R. (1995, April). A performance model for real-time systems. *Naval Air Warfare Center, XV*, 59-60.

The work of Dr. Newport presents the differences of artificial scenarios benchmark to that of real scenarios in which the actual application code is used as the tool to drive the benchmark. His paper explains how difficult it is to simulate a benchmark environment based on imply layers of software effects due to the coupling action of software such as firmware, runtime system, operating systems, etc. Dr. Newport recommends that the embedded system analyst use actual application code as a benchmark that inherently performs IO and interrupts without recurring to artificial execution scenarios. His work relates specifically to benchmarks on ADA systems.

Null, L. & Lobur, J. (2003). Chapter 10. In *The essentials of computer organization and architecture* (pp. 10-11). Boston: Jones and Bartlett Publishers.

A brief description of the tools and techniques required for performance benchmarks. This chapter provides the necessary knowledge to correctly interpret performance benchmarks results and avoid misreporting and misinterpretation of results by using the appropriate statistical tools and formulas.

Saavedra,R. H. & Smith,A. J. (1996).Analysis of benchmark characteristics and benchmark performance prediction. *ACM Transactions on Computer Systems*, *14*, (4) 344-384.

This article presents a benchmarking model based on a set of abstract operations representing a particular programming language basic operators and language constructs that appear in programs. Through a machine characterizer program, the machine characteristics are obtained to calculate the execution of each abstract operation. The purpose of this model is to present not only the results of the benchmark but also provide enough information to be able to explain the reason for such a result.

Staelin, C. (2005). Lmbench: an extensible micro-benchmark suite. Retrieved July 1, 2006, from www.interscience.wiley.com.

This paper described the use and design of lmbenchmark. Lmbenchmark is an extensible micro-benchmark suite, the micro-benchmarks include bandwidth: file re-read using read() and mmap(), IPC communication using TCP, pipe, and Unix sockets. It also measures latency; memory latency, TCP and UNIX socket connection, IPC communication using TCP, UDP, RPCs, pipe and UNIX socket, file creation and detetion, process creations using fork(), fork()+exec(), and sh(), select() on file descriptors and network sockets, etc.

Vokolos, F. I. (1998). Performance testing on software systems. In Elaine J. Weyuker (Ed.), *First international workshop on software and performance* (pp. 80-81). Sante Fe, CA: ACM.

The paper discusses approaches to software performance testing, performance testing objectives including the design of test cases selection or generation algorithms to test the performance criteria, the definition of metrics to assess the comprehensiveness of a performance test case, and the comparison of different hardware platforms for a given application.

Wasserman, T. J. & Martin, P. (2004). *Sizing DB@ UDB servers for business intelligence workloads.* Ontario, Canada: Haider Rizvi IBM Toronto Laboratory,

The work in this paper describes the difference between performance testing and hardware sizing. It discusses the assumption that little or not system environment information is available for an specific workload and recommends sizing approach and set of steps to follow during hardware sizing activities including high-level input data from customers, cross-check input data estimates, system resources demands for each workload.

Weyuker, E. J. & Avritzer, A. (2002). A metric for predicting the performance of an application under a growing workload. *IBM Systems Journal, 41* (1) 1-5.

This paper presents a new metric, Performance Nonscalability Likelihood (PNL) to predict whether a software system will provide satisfactory performance under workload increase. The method relies on the collection of workload statistics to characterize the target system, then classifies the different aspects of the workload and the use of a performance testing plan to obtain response time and CPU cost per transactions.

White paper: Client and server system performance benchmark. (2004,April). Retrieved July 1, 2006, from http://whitepapers.zdnet.co.uk/0,39025945,60085803p-39000531q,00.htm.

This paper describes basic benchmark guidelines to be observed when conducting hardware benchmarks. These guidelines are based on the approach implemented by benchmarking entities such as Standard Performance Evaluation Corporation (SPEC),
Transaction Processing Performance Council (TPC) among others. The paper describes the objectives of performance benchmarks, and basic observations such as understanding the benchmark, being aware of system optimizations and the avoidance of orange-to-apple comparison.

Whitman, M. E., & Woszczynski, A. B. (2004). *The handbook of information systems research.* San Francisco: Idea Group Publishing.

In this chapter, Theresa M. Vitolo, Gannon University, USA and Chris Coulston Penn. State University, USA Explore the reasons why simulation techniques has not been fully embraced in IT as a research tools as opposed to fields like physics and social sciences. This chapters provides insides about the different ways in which the IS research community can benefit from simulation tools and techniques in IS research. Considering simulation as a possible tool to provide a solution to the IS problem addressed here.

Wong, A. T., Oliker, L., Kramer, W.T.C., Kaltz, T. L., & Bailey, D. H.. (2002) ESP: A system utilization benchmark. national energy research scientific computing center. Berkeley, CA: Lawrence Berkeley National Laboratory.

This article describes the Effective System Performance (ESP) benchmark framework. ESP is designed to measure system-level performance including shutdown-reboot time, job scheduling

efficiency and the ability to handle large jobs. The article explains how traditional benchmark tools such as LINPACK and NAS Parallel Benchmark report sustained computational performance for individual jobs, but lack information about system utilization.

Zhang, X. (2001) *Application-specific benchmarking*. Unpublished doctoral dissertation., Harvard University, Cambridge, Massachusetts.

In this thesis work, Zhang presents techniques for designing meaningful benchmarks and how to incorporate application specific characteristics into the benchmarking process. The paper is based on Hbench benchmarking infrastructure and its superiority in predicting application performance. Hbench implements vector-based benchmarking methodology for which advantages and disadvantages are discuss in the paper.

Zhang, X., & Seltzer, M. (2000). *Hbench:Java: An application-specific benchmarking framework for java virtual machines*. Cambridge, MA: Harvard University Press.

A vector-based benchmark framework for Java Virtual Machine paper; the paper presents the goals of the benchmark, to compare the performance and to reason about why applications run faster on one system than on another, to guide performance optimizations, and to predict an application's performance on nonexistent platforms.

References

A guide to the project management body of knowledge (3rd ed.). (2004). Retrieved July 24, 2006, from www.pmi.org

Balbo, G. et al. (2000). Introductory tutorial petri nets. In K Jensen & J. Desel (Ed.), *21 international conference on application and theory of petri net* (pp. 1-169).. Retrieved July 14, 2006, http://www.informatik.uni-hamburg.de/TGI/PetriNets/introductions/pn2000_introtut.pdf

Cassier, P. (2005). *Effective zseries performance monitoring using resource measurement facility*. Armonk, NY: IBM Red Books.

Cvetanovic, Z. et al. (1990). Perfect benchmarks decomposition and performance on VAX. In Dileep Bhandarkar & Tom Barday (Ed.), *Proceedings of the 1990 Conference on Supercomputing* (pp. 455-474). Retrieved July 18, 2006, http://portal.acm.org/citation.cfm?id=110382.110463

Darema, F. (2001). Performance evaluation with real applications. In Rudulf Eigenmann (Ed.), *Performance evaluation and benchmarking with realistic applications* (pp. 4-5). Boston: MIT Press.

Dujmovic, J. J. (2001). Universal benchmark suites - A quantitative approach to benchmark design. In Rudulf Eigenmann (Ed.), *Performance evaluation and benchmarking with realistic applications* (Chap. 6). Boston: The MIT Press.

Foltier, P. J. & Michel, H. E. (2003). *Computer system performance evaluation and prediction*. Burlington, MA: Digital Press.

Freedman, D. H. (2006, January). What's next. *Inc, 28*(1), 57-58

Gertphol, S. (2003). MIP formulation for robust resource allocation in dynamic real-time systems. In IEEE Computer Society (Ed.), *International parallel and distributed processing symposium (IPDPS'03)* (p. 17). Los Angeles: Author.

Gibbs, G. B. et al. (2004). Chapter 1. In *IBM eservers series capacity planning: A practical guide* (p. 1.7). New York: IBM Redbooks.

IBM Compiler Optimization Argument Example. (2004). Retrieved July 18, 2006, from http://www.nersc.gov/nusers/resources/software/ibm/opt_options/optex.php

King, B. (2005). *Performance assurance for IT systems*. Boca Raton, FL: Auerbach Publications.

Lilja, D. J. (2000). *Measuring computer performance: A practitioner's guide*. Minneapolis: Cambridge University Press.

LINPACK. (1970). Retrieved July 18, 2006, from http://www.netlib.org/linpack/

McKenzie, J. (2000). *Beyond technology: Questioning, research and the information literate school*. Bellingham, WA: FNO Press

Mohamed, A. (2006). *Companies move from batch processing to real-time data feeds on IBM mainframes*. Retrieved July 10, 2006, from http://www.computerweekly.com/Article137548.htm

Newport, J. R. (1995, April). A performance model for real-time systems. *Naval Air Warfare Center, XV*, 59-60.

Null, L. & Lobur, J. (2003). Chapter 10. In *The essentials of computer organization and architecture* (pp. 10-11). Boston: Jones and Bartlett Publishers.

Petri, C. A. (1962). PetriNet. Retrieved July 14, 2006, from http://en.wikipedia.org/wiki/Petri_net

Petri Net. (2006). Retrieved July 31, 2006, from http://en.wikipedia.org/wiki/Petri_net

Rabb, F. et al. (n. d.). *Overview of the TPC benchmark C: The order-entry benchmark*. Retrieved July 19, 2006, from http://www.tpc.org/tpcc/default.asp

Ranadivé, V. (1999). *The power of now: How winning companies sense and respond to change using real-time technology*. NY: Author. (Original work published 1999).

Riedl, R. (2001). Need for trace benchmarks. In Rudulf Eigenmann (Ed.), *Performance evaluation and benchmarking with realistic applications* (Chap. 6). Boston: MIT Press.

Seltzer, M. & Krinsky, D. (1999). The case for application-specific benchmarking. In IEEE Computer Society (Ed.), *The seventh workshop on hot topics in operating systems* (pp. 100-102). Boston: Author.

SPEC SFS Suite. (1995). Retrieved July 19, 2006, from http://www.spec.org/osg/sfs93/

The Expect Home Page. (2006). Retrieved July 26, 2006, from http://expect.nist.gov/

The Standard Performance Evaluation Corporation (SPEC). (1995). Retrieved July 18, 2006, from http://www.spec.org

Unix Top. (2006). Retrieved July 26, 2006, from http://www.unixtop.org/

Using HP PerfView. (2006). Retrieved July 26, 2006, from http://h20338.www2.hp.com/hpux11i/cache/325436-0-0-225-121.html?jumpid=reg_R1002_USEN

Vokolos, F. I. (1998). Performance testing on software systems. In Elaine J. Weyuker (Ed.), *First international workshop on software and performance* (pp. 80-81). Sante Fe, CA: ACM, NY.

Wikipedia. (2006). Retrieved August 1, 2006, from Wikipedia online Encyclopedia Web site: www.wikipedia.com