Engineering Management
Field Project

Hardware Sizing for Software Application

By

Ganesh Swaminathan

Spring Semester, 2009

An EMGT Field Project report submitted to the Engineering Management Program and the Faculty of the Graduate School of The University of Kansas in partial fulfillment of the requirements for the degree of Master's of Science.

____
Tom Bowlin
Committee Chairperson

____
Herb Tuttle
Committee Member

____
Ray Dick
Committee Member

Date accepted:______________________________

Acknowledgements

My sincere thanks to Dr. Tom Bowlin, Engineering Management (EMGT) Program advisor and the chairperson of the advisory committee, for his guidance throughout the project.

I also want to thank Prof. Herb Tuttle and Ray Dick for their valuable time and feedback in reviewing the project work and helping me with the organization of the material. The EMGT study has been the best investment of my life. I have enriched my knowledge in so many different areas. It has been a wonderful experience working with all the faculty members. I like to extend my gratitude to all of the EMGT faculty members for their efforts in making the EMGT Program enjoyable and successful.

Executive Summary

Hardware sizing is an approximation of the hardware resources required to support a software implementation. Just like any theoretical model, hardware sizing model is an approximation of the reality. Depending on the infrastructure needs, workload requirements, performance data and turn around time for sizing, the study (Sizing or Capacity Planning) can be approached differently.

The most common method is to enter all the workload-related parameters into a modeling tool that is built using the results of workload simulation on different hardware. The hardware and software requirements are determined by the mathematical model underlying the tool. Without performing a test on the actual hardware environment to be used, no sizing can be 100% accurate. However, in real-life there is a need to predict the capacity when budgeting hardware, assessing technical risk, validating technical architecture, sizing packaged applications, predicting production system capacity requirements, and calculating the cost of the project. These scenarios call for a quick way to estimate the hardware requirements. When dealing with prospects, there is a need to come up with credible and accurate sizing estimates without spending a lot of time.

One of the challenges faced by Kronos is the amount of effort and time spent in hardware sizing for prospective customers. Typically, a survey process collects the workload related parameters and feeds the sizing tool, which uses the performance model based on benchmark test results to produce the hardware recommendations. Although this process works great for customers, it is a time consuming activity due to the collection and validation of large number of independent variables involved in the current sizing model.

This project makes an attempt to delve into alternate methods for producing quick sizing. By combining the empirical data collected from various production systems and simple statistical technique, relationship between sizing factors and CPU rating can be established. This can be used to create a simple model to produce a quick, easy and credible recommendation when sizing new customers.

Table of Contents

List of Figures

List of Tables

Chapter 1 -- Introduction

Kronos provides an integrated suite of human resources, payroll, scheduling and time and labor solutions to companies of all sizes, across a variety of industries. Workforce Solution helps organizations hire and manage people most effectively by maximizing the productivity while minimizing the impact on IT. The integrated suite includes almost 15 different modules of applications that can be activated based on the licensing. The company has more than 40,000 customers world-wide. The diversity of customer environment, business applications implemented, and the hardware alternatives available to meet the demand, results in a significant variation in hardware requirements among the customers. One of the key factors to successful implementation is configuring adequate hardware capacity so that the system performs well. Sales, Service, Support and Engineering are the four major divisions in the company. There are well defined processes that guide a new project from sales to service and support.

One of the challenges identified was the significant amount of unfunded work performed by the professional services organization in completing the hardware sizing for new customers. The focus of this project is to examine the sizing methodology and the elements considered when developing a hardware recommendation for a new customer and suggest ways for improvement. The ultimate goal is to simplify the process, thereby eliminating or significantly reducing the amount of unfunded work.

What is Hardware Sizing?

Sizing is an approximation of the hardware resources required to support a specific software implementation. The input for this process is a set of workload-related parameters and the output is hardware specification in terms of processing power, storage, and memory requirement for optimal performance of the application.

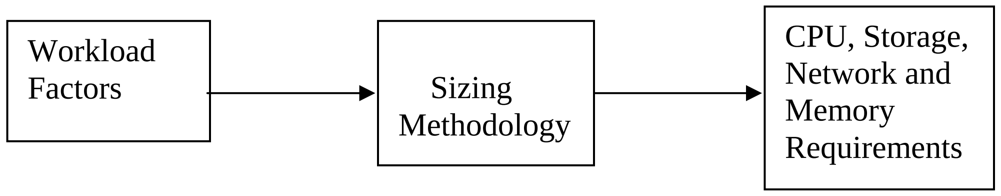


Figure 1: Sizing

The processing power is broken down for hosting each component like the Web Server, Application Server and the Database Server.

Customers need confidence in answering fundamental performance questions, such as the following:

- Will the system have good responsiveness, particularly during peak periods?
- Will there be enough capacity for expected growth and variations in use?
- What hardware alternatives are available to meet the expected demand?
- Will the hardware investment be utilized effectively?
- Will application scale adequately for the business?

One of the key factors to successful implementation of a software product is configuring adequate hardware capacity so that the system performs well. The sizing methodology employed should effectively address all performance related questions. Hence, most sizing methodology assumes a thorough knowledge of the application and follows the approaches taken for sizing with capacity planning.

Challenges of Sizing for a New Customer

Almost all software vendors face these challenges when it comes to estimating hardware for a new implementation. The biggest reason is the limited information that is available during the early stage of the sales or the planning cycle. Most vendors including Kronos have developed a sizing questionnaire. During the sales process, a questionnaire or survey is completed by the prospective customer, which is then used to estimate the hardware resources required for the project. The questionnaire aims at collecting information on typical characteristics of the product usage by the customer. The estimated hardware resource is used by the customer to arrive at the total cost for implementing the product.

The problem with the survey is that many of the answers are either unknown or is a very rough estimate. Often these questions are not even answered by the right team. When you start a sizing study, there are never enough solid facts. The known facts are vague. This presents a moving target as people change their mind or constantly review and modify the facts. For example, questions like how many reports will be running during the busy times or how many concurrent users will be on the system are important criteria that affect the hardware sizing but these questions may not make much sense for a new customer. Hence, there is a risk that information collected during the survey can be too conservative or aggressive.

Inaccurate information when fed into the sizing model produces a not so accurate estimate of hardware requirements. This poses a challenge later during the implementation cycle when the estimates are modified. One way many vendors overcome this challenge is by setting the right expectation of the sizing process upfront. Sizing is generally positioned as an iterative process that will be refined and repeated several times during the course of the implementation as more information is available. Though this process works for many but it does not flow very smoothly. The cost of project is calculated way upfront during the sales cycle and the ROI (Return on Investment) calculation includes the cost of the right hardware for the product. Any significant change in the hardware estimates can make the project exceed the allocated budget. So the initial sizing process has to make reasonable assumptions about the use of the product and produce fairly accurate sizing requirements.

At Kronos, the sales team works closely with the professional service team to produce the sizing recommendations. The information collected during the survey is analyzed by technology professionals. There are several back and forth conversations before the information is analyzed by the sizing model. The final hardware recommendations document is provided to the sales team. In other words, there is a significant service involvement and these hours are usually unfunded.

Problem Definition and Scope

Following are the challenges in the current sizing methodology for new customers –

- The criteria collected during the initial survey from the new or prospective customer is not very accurate

- More time is consumed by the technology professional in making reasonable assumptions and producing a sizing recommendation

- The work performed by technology professional for sizing is not funded.

A simplified and yet accurate sizing process that can be executed by the sales team with very little help from the service organization is needed to minimize the unfunded hours. The primary goal of this project is to research the possibility of a simplified approach for hardware sizing by studying the current sizing model and application usage characteristics. The expected outcome is a model that is simple to understand and simple to use, thereby reducing the effort in collecting numerous variables to determine the processing requirements.

The scope of the project includes only the Time Management, Leave and Attendance module within the suite of applications. The sizing process will focus on estimating one type of resource – CPU power – needed for optimal performance.

This project requires an understanding of the science behind hardware sizing, current sizing methodology, architecture and the usage characteristics of the application. Although the problem is specific to Kronos, the methodology and the solution discussed in this project can be applied to any hardware sizing scenario.

Application Architecture

Workforce Central is a 3-tier J2EE application. Users connect using the browser to a middle tier, which is a web/application server. The application deployed on the middle tier connects to the database server to retrieve the data. Except for data validation most processing is done on the middle-tier.

At the heart of the architecture is the totaling engine, a multithreaded servlet deployed within the J2EE framework. Multiple applets are deployed within the application to allow editing and retrieval of data from a back-end database. Application can be deployed on multiple application servers for high availability and load balancing.

A given environment can have several application servers depending on the user load, hardware available and the failover requirements, and one database server.

Current Sizing Methodology

The current hardware sizing methodology employed follows these steps:

- Build use cases—quantify how customers use the application
- Determine the performance characteristics of application—Quantify performance measures
- Build hardware sizing methods — Provide hardware sizing tools

The first step in estimating the hardware required for a Workforce Central implementation is to understand how a customer will use it. The goal of this exercise is to quantify use cases in a way that can be tested and then modeled.

Use Cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. They are a simple and powerful way to express the functional requirements, or behaviors, of a system. Use Cases provide a way to express a system's requirements.

From a sizing perspective, a use-case is a set of workload conditions that represent a pattern of functional usage of the product. Different use-cases are identified that result in a significant workload. Once the use-cases are identified, then a possibility of overlap of these use-cases is considered. The hardware is sized for the heaviest use-case; one that requires most significant amount of resources. The general sizing philosophy is to ensure that available processing power exceeds the heaviest workload requirement by 50%. This allows for unexpected spikes in the usage and also allows the system to scale in the future.

For new products and features, the Kronos product management group establishes use patterns and response time requirements for all the product features. Product management uses customer interviews, review boards, expert industry consultants, and data gathered during on-site visits.

For existing products, customer data is collected and analyzed by product management and performance engineering groups. Use-cases are then modified as more experience is gained with those products across industries. Because the manner of use is generally the primary driver in hardware sizing, understanding customer use-cases is a continuing activity that results in improving accuracy of hardware estimation throughout a product's life cycle.

When establishing use-cases, it is necessary to determine the timing of actions each user takes and then determine the concurrency across all predicted users over a given time.

Below is an example of Pay Period Use-Case - a typical activity that occurs during the end of Pay Period.

> "*End of pay period processing occurs in a two-hour time window on Mondays, between 2:00 P.M. and 4:00 P.M. The arrival distribution of managers is a normal distribution, and two standard deviations complete their edits within the window. Supervisors login to application to find the employees who require corrections to their timecards. Managers select the employees requiring corrections and for each employee, they enter the Timecard editor. The necessary edits are performed, at a transaction rate of two edits per minute, and totals are calculated for each change.*
>
> *For Hourly Employees, each manager:*
>
> *Corrects 25 late entries.*
> *Corrects 5 employees who have not approved.*
>
> *For Hourly employees, each manager:*
> *Selects the 5 employees who need correction.*
> *Corrects 5 missed punches.*
> *Corrects 5 employees who did not approve.*
>
> *For Project employees, each manager:*
> *Selects the 5 employees who need correction.*
> *Corrects 5 employees who have less than 40 hours.*
> *Corrects 5 employees who have forgotten to approve.*

> *Finally, each manager runs Exceptions and Time Sheet reports against the previous pay period, with a 60-second wait between the completion of a report and the submission of the next report (or logoff)*".

Once the use cases to be measured are known, their impact on system performance can be identified. In some cases, the use-cases are non-overlapping (that is, they do not occur during the same time in the course of a normal pay period), and overall scalability is determined simply by evaluating the single case that places the greatest load on the system.

In other cases, the overlap of two or more functions requires a test methodology that takes concurrent load from those functions that overlap and models them to reflect their impact on the system.

The next step in the sizing methodology is to define the performance characteristics of the application. The application-specific measures of performance used are:

1. Response Time - the time duration between a request and a response; for example, the time required to display a new screen after a "back" button is pressed.

2. Throughput - the number of requests within an interval of time of a given transaction that the system can fulfill with given hardware configuration.

3. Resource Utilization - the quantity of computing resources required at a given throughput or load condition.

Two basic kinds of tests are constructed to measure system performance.

1. Single-user tests - single-user tests exercise the operations done by a single user with no concurrency. This test establishes a baseline for response times against which system scalability can be determined when concurrent users are added. Single-user tests are also used to test the breadth of features across a product. All significant user operations are generally tested on a single-user basis.

2. Multi-user tests - multi-user tests are used to measure the computing resources used when increasing load is applied. The primary computing resources measured are CPU utilization, memory consumption, hard drive input/output rates, and network utilization. Response time for various activities is captured. Multi-user tests also provide a measure of system throughput capabilities.

Both single-user and multi-user tests are generally automated so that results (metrics collected) are repeatable. Many repetitions of tests on both similar and varied hardware

are run to collect data for the sizing models. The results of the benchmark tests are used to develop a performance model.

The final step is to build an algorithm or model that computes the quantity of system resources consumed when the product is used and configured in a particular way. The way a product is used and configured is modeled by defining a set of independent variables. Independent variables are quantifiable inputs or configuration parameters for a system.

Examples of independent variables include the following:

- The number of employees using the system
- The number of supervisors using the system
- The number of configured pay rules
- The database platform used
- Modules implemented
- Number of concurrent users
- Number of reports run

Analysis and independent variable testing, also called sensitivity testing, is performed to determine the effect of various independent variables on the consumption of system resources. Examples of system resources are CPU computing power, RAM memory, hard drive memory, and network bandwidth. The consumption of these resources is known from the results of the product performance and benchmark tests. A hardware sizing tool is developed from the performance model and hardware choices are incorporated into the tool. The tool is built to produce the hardware recommendations with a given set of workload parameters based on the performance model. Appendix A lists most common independent variables used by the sizing tool to produce hardware recommendation.

Flow of the current sizing process

Use-cases collected from the customers are based on a survey. The survey includes a variety of questions about the customer's environment, hardware purchase standards, support requirements and the proposed product usage. The service organization generates a "Site Survey" form that is used to collect data about prospective implementations. This survey provides the assurance that a consistent methodology is used to collect data necessary for a good sizing estimate. Usually, a Kronos technology professional fills out a site survey during a series of interviews with a customer. From a completed site survey, the consultant uses the hardware sizing tool to determine a hardware sizing estimate and presents the results in writing to the customer. This process can be completed in the range of a few days for a simple implementation to a number of weeks for larger installations with complex requirements.

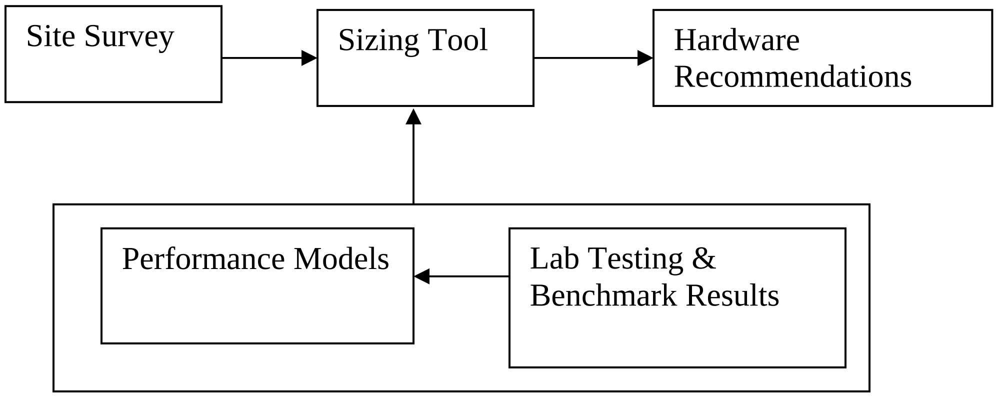


Figure 2: Current Sizing Process

Sizing tool uses performance models established internally based on a variety of testing with different use-cases; model is constantly validated with customer data collected from live sites as well as test results from benchmarks run at independent laboratories on a variety of hardware platforms.

Chapter 2 – Literature Review

Practical Sizing or Capacity Planning is a predictive process to determine the computing hardware resources required to support a given workload or estimated changes in future workload. As a result of predictive nature, capacity estimates can only be an approximation at best. There are several white papers that talk about the overall approach for sizing the hardware.

One of the problems is the confusion between the terms ‘Sizing' and ‘Capacity Planning’ (Larry Pedigo, 2004). Sizing is best described as the process of estimating the hardware requirements of a planned application based on technical descriptions of the customer’s needs. Capacity Planning is the process of measuring resource requirements from existing applications, and projecting the amount and kind of hardware required to support larger workloads.

According to Larry Pedigo (2004), most of the available literature on ‘Sizing’ assumes a thorough knowledge of application. Most white papers and articles on ‘Sizing’ are actually focused on capacity planning. In a brand new project, before the first server is purchased, customer has no idea what resources will be required by the new application. Capacity planning guidelines will be of little use to the customer. The focus is more on hardware sizing based on ‘rules of thumb’ and guidelines based on the customer’s technical needs. According to the authors of the e-book on ‘IBM @Server pSeries Sizing and Capacity Planning (2004)’, capacity planning is strongly related to sizing. Capacity Planning is part of the resizing task that happens when the actual performance data from the production system is used as input for sizing.

The article "The Ratio Modeling Technique (1997)" by David Cook, Ellen Dudar and Shallahamer Craig introduces a new calculation based method for performing capacity predictions. Author defines ratio modeling technique as set of steps to perform a quick sizing prediction. This technique shows an alternate method to predict the production system load based on relationship between process categories (e.g. Batch processes) and system resource (e.g. CPU). This paper shows how a simple technique can be used for real situations that demand predicting capacity requirements at a low precision level. Author also quotes that this technique has been used and validated by many of Oracle’s largest customer sites.

The author Shallahamer Craig in another article “Predicting Computing System Throughput and Capacity (2002)” presents an overall project plan to conduct a capacity planning study and also defines three different study types. The different predictive study types all revolve around the concept of precision and statistical confidence. Depending on the level of precision required and the investment (time and effort) available, a capacity study can be organized as an initial performance readiness review or a limited performance assurance test or a classic performance assurance test. The three approaches differ in the data inputs and the methods used for study. Author goes on to discuss different methods - Estimation, Summation and Process method to produce capacity prediction at varying confidence levels. In Summation method, author suggests Regression Analysis on metrics gathered from production data to produce a fairly high confidence model.

Available literature on sizing often falls short on tools needed to perform the sizing estimates. Regression analysis is a simple method for investigating relationships among variables. The book “Regression Analysis by Example (2006)” by Samprit Chatterjee and Hadi Ali provides essentials of regression analysis through practical applications. While this book provides only a review of the basic principles of regression, it covers many other topics, mainly problems occurring during a regression analysis. Diagnostic tools as well as methods to overcome the problems are discussed. Throughout the book there are plenty of examples to demonstrate the ideas presented. Examples are derived from a wide range of disciplines and present real problems. In summary, this book contains a thorough overview of diagnostic tools for regression models that are easily understood.

The White paper "Sun Server Scalability and Sizing Guide (2002)" from Sun Microsystems, Inc. provides a simple formula or method of sizing Sun Enterprise server based on benchmark testing results. This paper shows how benchmark results can be used for predicting CPU, Memory and disk capacity configuration requirements.

The White paper "Windows Sever 2003 Terminal Server Capacity and Scaling (2003)" from Microsoft contains analysis, results, sizing guidelines, and testing methodologies for the Terminal Services component in the Windows 2003 Server family. This methodology specifies profiles for Light, Medium and Heavy users, allowing the vendor to establish performance characteristics for these three classes of users. A model is created based on the user types to arrive at the server sizing requirements.

Larry Pedigo in his white paper “Sizing Oracle on Microsoft Windows and Dell PowerEdge Servers (2004)” defines a systematic approach to sizing. The key steps outlined in this paper are:

- Define customer requirements
- Collect customer information
- Analyze system data
- System determination
- System test (optional)
- Implement

In system determination step the technical data gathered is extrapolated to design a system that will meet the customer requirements. The model is used to predict total CPU usage, total memory usage, and total I/O load. At this step, all the available sizing guidelines including any relevant benchmarks and “rules of thumb” are pulled together.

"How to Effectively Size Hardware for your Portal implementation (December 2004)" by Jason Pepper, Jack Sun, and Biswajit Nayak is an Oracle White Paper article that explains the overall capacity planning methodologies available for Oracle Portal application. Though the calculations and the metrics obtained are more specific to the portal application, the overall sizing and the estimation methodology used can be applied to any application.

There are three primary approaches for sizing hardware for a software implementation.

- Algorithm based
- Example based
- Pilot based

Regardless of the approach, a model is needed for predicting the capacity. Modeling is an integral part of planning capacity (Shallahamer Craig, 2002). Models can be broadly classified into either simulation models or mathematical models. Mathematical models require input from either a production system or a simulated production system. If mathematical models are being used, unless the observation is from a production system, some form of workload simulation must occur to feed the mathematical models. Constructing simulation and mathematical modeling tools can be very difficult and time consuming. Determining which model or models to use is dependent upon the method chosen and the data available.

Algorithm Based

In this approach, an algorithm or process accepts input from the customer (e.g. user counts, number of reports run, total page requests, number of transactions, etc.) and attempts to deliver a processing requirement. This is probably the most commonly accepted tool for delivering sizing estimations. Unfortunately, this approach is generally the most inaccurate. When considering a logical n-tier enterprise class implementation, the number of variables involved in delivering a calculation for realistic sizing requires numerous variables that calculations become so complex and sensitive. A small variation in input variables can produce inaccurate results. The majority of the vendors use some form of algorithm based approach to recommend hardware for a new implementation.

Benchmarking the application and modeling the data from the testing is probably the most common approach in arriving at an algorithm. A benchmark is defined as a set of programs that are run on different systems to give a measure of their performance. The term performance becomes very crucial and needs to be defined before the testing. In order to understand what needs to be measured during the testing, a good understanding of application architecture and the usage characteristics is absolutely essential. Performance metrics are defined and tools are developed to trace and record the metrics during the testing. Appendix B lists some of the popular metrics used by various application benchmarking.

There are two kinds of benchmark testing – standard and application specific. A standard benchmark usually includes a fixed set of programs that are run on different systems to produce a single figure called rating, which is then used to rank system performance. This type of testing has little relevance when the characteristics of the standard programs are different from the real application. Even in cases where the application characteristics match the benchmark programs, many take a cynical view the results.

Several standard institutes are in charge of the process of defining and distributing standard benchmarks. Some of the well-known standard entities are: SPEC (System Performance Evaluation Corporation) and TPC (Transaction Processing Performance Council). SPEC defines a wide variety of standard benchmarks, ranging from high performance computing to network file servers. Among those, the SPEC CPU benchmark suite is probably the most widely used benchmark in computer literature. SPEC is a non-profit corporation formed to establish, maintain and endorse a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC develops benchmark suites and also reviews and publishes submitted results from their member organizations and other benchmark licensees.

SPEC publishes results for most widely used CPUs in the industry. Below is an example of SPECInt 2006 results.

Company	System Name	#CPU	Rating
Dell	Power Edge 3250 (1.4GHz/1.5MB, Itanium2)	1 core, 1 chip	824
Dell	Power Edge 7250	1 core, 1 chip	1403
HP	ProLiant DL380 G4	1 Core 1 Chip	1433

A rating like this helps to compare different systems. However, it should be noted that the rating was developed using the standard set of programs. There are different suites of programs used for testing. SPECInt is a rating based on integer suite. There is one called SPECFloat, which is based on the Floating-point suite. SPECMail is a benchmark for Mail Server programs. Depending on the characteristics of the real application, a particular rating method can become more relevant over other ratings.

The other type of benchmarking is more application specific and is used by most software vendors to produce some form of hardware sizing recommendation. There are several different approaches, methods and tools available to conduct this type of testing. This also suggests that many believe that systems should be measured in the context of applications in which end-users are interested. The basic idea is to separate characterization of an application from that of the underlying platform and combine the two characterizations to form a prediction of the application's performance. As the application of interest is incorporated into the "benchmarking" process, the resulting performance metrics reflects the expected behavior of the application on the given platform. Whatever is the approach or the tool used vendors, they all embrace the following steps.

- Define Workload Characteristics
- Define Performance Targets
- Conduct Simulation
- Define & Collect Metrics
- Develop Performance Model
- Develop Sizing Tool

Defining workload characteristics involves understanding the real-life application usage patterns and developing meaningful use cases for benchmarking. The workload tested should reflect the actual customer usage as closely as possible. Workload information is very crucial when publishing the benchmark results. Customers would want to know how the application performed given a particular scenario or workload. From a vendor standpoint, this helps to build the confidence of the customers on the performance characteristics of the application.

Specific performance targets are established during the benchmarking process. For example, Order Entry response time under three seconds or maximum CPU utilization allowed is 80%. Measuring against specific performance goals also helps to verify whether the application meets the designed performance expectation. If the application does not meet the designed goal, corrective action is usually taken. A series of tuning or optimization steps are carried out and the focus is on identifying and eliminating bottle necks to improve performance. Sizing or performance factors emerge during this iterative testing process. For example, the number of images of 20k size extracted is around 20,000 in three seconds and the time increases if the average size of the images is around 40k, then the average size of the image becomes one of the sizing factors. Several performance terms are used. Appendix B lists some of the most common terms.

The principle of a benchmarking is to simulate the behavior of real users with "virtual" users. The Simulation tool is used to record the behavior of the application under the load and give information on the virtual users' experiences on a given hardware.

Simulation software is often distributive in nature. It is applied on multiple servers running simultaneously, with each server simulating multiple virtual users. Ongoing records of the virtual users' experiences during the tests, including response times and errors are maintained. Focus is more on the performance of the application and the database software under stress. Both Baseline (Single-User) and Scalability (Multi-User) tests are carried out.

The following table (Table 1) shows sample results from a multi-user test exercising a use-case. These tests were run to determine scalability characteristics. The 50 concurrent users test was run with a single web application server; the 150 concurrent users test was run with three web application servers. As can be seen from the results below, the application demonstrates excellent scalability. Response times were measured at the client. All response times are shown in seconds.

Total Users Processed per Hour	157	466
Peak Concurrent Users	50	150
User Action 95th Percentile Response Times		
User Session Time	1,148	1,159
Logon Action	1.141	1.828
Next Applet	.27	.36
Save Applet	.761	.969
Report for 5 employees	.03	.032
System Performance Data		
Database server CPU utilization	14.03%	47.01%
Database server CPU seconds/user	6.44	7.26
Database server network bytes/second	356,891	963,156
Web server CPU utilization (per Web server)	45.77%	46.49%
Web server CPU seconds/user	21.02	21.55
Web server network bytes/second	288,619	282,174

Table 1: Sample Benchmark Results

To provide a serviceable calculation model when benchmarking concludes, it is important to identify the factors that affect the load on a given system. Before recording any tests, several iterative tests are performed to provide feedback on the health of the system. Based on the measurements of these iterative tests, several tuning exercises are carried out within both the middle-tier and the infrastructure. Tuning or the optimizations done are recorded and usually form the best practices for the application environment.

Sensitivity testing is performed to determine the effect of the factors on the consumption of system resources. Examples of systems resources are CPU computing power, RAM memory, hard drive memory, and network bandwidth. Examples of the sizing factors are number of concurrent users, number of concurrent page requests, number of documents stored in the database, product features used etc. A performance model is built based on the correlation between the sizing factors and the resource consumption.

Testing is performed on a finite number of hardware with varied specifications (like two, four, eight CPUs) from different vendors or operating environments. Generalization rules are applied for all other hardware and specifications. This is typically done by the sizing tool. Here is an example of how the sizing tool might use the performance results to arrive at hardware requirement.

1000 users of a sales and distribution department and 1000 users of a finance department plan to use the ERP system on 400 MHz systems. For performance and security reasons (for instance, head room for batch jobs), a maximum CPU utilization of 67% is requested by the customer.

First the tool maps the "Finance" users to "Sales and Distribution" users, so we get 500 "Sales and Distribution" users instead of 1000 "Finance" users.

The scalability tests have shown that a 300 MHz CPU can handle the load of 40 "Sales and Distribution" users. Therefore, a 400 MHz CPU handles 50 users, and 1500 users need 30 CPUs. Using the customer requirement for 67% CPU utilization, tool configures a system containing 45 CPUs.

The software configuration consists of the database and the ERP system. Both components could also run on separated machines. To configure them, the tool uses the ratio of the CPU consumption of both the database and the ERP system (33% to 66%) determined through the Scalability Tests, and adds one CPU on each machine to handle the network traffic. As a result, we configure 16 CPUs for the database and 31 CPUs for the ERP system.

Example Based

An Example-based approach requires a set of known samples to use as data points along the thermometer of system size. The more examples available, the more accurate is the sizing. By using samples collected from real world and through internal deployments, customers can be assured that the configurations proposed have been implemented before and will provide the performance unique to the proposed implementation. The challenge is with maintaining an accurate database of real-world hardware configurations and determining configurations that produce optimal performance.

Pilot Based

A proof of concept or pilot based approach offers the most accurate sizing data of all three approaches. Performance metrics is defined and collected from the customer environment during the pilot implementation. Number of users and the activities are simulated on the hardware provided by the customer. Based on the results, the exact processing requirements are arrived. By far this is the most expensive and most time consuming method. Moreover, this approach requires the customer to have manpower, hardware, and the time available to implement and validate the solution. Findings are analyzed based on the test results. Because the environment is simulated as close as possible to the real solution, the results are highly accurate.

Chapter 3 - Procedure and Methodology

The objective is to minimize the labor effort involved in the sizing for a prospective customer at the same time produce a reliable sizing recommendation using the known information. Majority of the efforts expended during the current sizing process is directed towards collecting, validating and making reasonable assumptions about the independent variables that affect the required processing power. While the current process is labor intensive, it is a proven and well established methodology with very high credibility.

The current sizing process produces recommendations based on several independent variables collected during the survey. Below are the most common independent variables collected for sizing the Workforce application.

- Peak Window duration
- Total Pay Period Employees
- Editing done by admins or individual managers
- Total supervisors
- Will totaling engine be used
- Database Platform
- Number of supervisors using HTML interface
- Use of Organizational Charts
- Will Accruals be used
- Number of Reports run per Supervisor
- Percentage of Employees paid from schedule
- Percentage of timecards edited daily
- Maximum supervisors creating schedules in a shift
- Number of employees scheduled by supervisors
- Peak scheduling window
- Maximum number of supervisors editing schedules in a single shift
- Total number of employees that will be using workflow
- Number of Time Collection Devices
- Number of employees punching during the Peak Shift
- Number of light users
- Number of Medium and Heavy Users
- Total number of workflows used

The approach taken in this project is calculation-based. However, instead of using a long list of independent variables and complex calculation, the overall philosophy is to present a simple to use and simple to understand algorithm based on empirical study. When all the independent variables cannot be accurately determined, it is not possible to perform high confidence capacity predictions. A quick and easy way to calculate hardware costs is needed when dealing with prospects for project budgeting, assessing technical risks and validating alternative solutions.

By analyzing the existing customer's data on the independent variables and the hardware used, a reasonable assumption about some of the lesser known variables can be established. Also, a simpler model can be developed by analyzing the correlation between the independent variables and the processing power used. Since the model is developed using the data from the real world environments, a high level of credibility can be established in the results.

Quantitative techniques like regression analysis, queuing theory, cluster analysis, probability statistics, and ratio modeling can be performed using the independent variables to predict the variation in the hardware power used by the customers. The dependent variable - CPU power - can be quantified based on SpecInt rating, a numerical value.

Following are the steps involved in this procedure –

- Identify the independent variables
- Collect Customer Data
- Convert the CPU resources used into SPecInt Rating (Numerical value).
- Analyze Data
- Use regression technique to identify correlation between the sizing factors and the CPU rating
- Develop a simple calculation for sizing
- Use customer data to validate the model.

Independent Variables

The first step in this process is to determine the factors or the independent variables that can affect the processing power (CPU) of the hardware. The factors identified in this step will dictate which data to collect from the customers. This is most crucial step because without the useful data, it is not possible to come up with a predictive model. Knowledge of the hardware sizing process, application architecture and experience with the current performance model are the essential ingredients for this step.

One of the problems with selecting independent variables for forecasting is the problem of collinearity – that is, severely correlated independent variables. If independent variables themselves are correlated then it becomes difficult to understand variation in processing power in relation to independent variables. Usually, if the variables are strongly correlated, whichever independent variable happens to be entered first typically accounts for most of the explainable variation in the dependent variable (Hildebrand and Ott, 1998). Also the standard error of the estimate can be very high with correlation variables. The objective is not to separate the predictive effects of every single variable but rather to arrive at a prediction equation. The model should have high predictability in explaining the variation with low standard error so it can be used with confidence.

Below independent variables can be determined accurately during the sales process. Customers usually know the products and features they are interested and how many employees and supervisors will use the product.

- Total number of employees in the system
- Total supervisors (or users)
- Modules that will be licensed
- Database Platform

Experience with the application has shown that number of concurrent users, number of reports run, and the duration of activity are some of the major factors affecting the utilization of the system. Besides the known variables, it is absolutely critical to collect data on the system usage to validate the workload characteristics and make reasonable assumptions about the unknown variables. Usage of the system with respect to the above variables and their correlation to resources used will be analyzed to understand the sizing needs.

Data Collection

This step involves collecting the below from a sample of customer sites –

- web and the application server log files
- hardware processing power used
- Number of licensed users, modules deployed and database used

Data gathered is never perfect. If the data is used without close examination, model predictions would be seriously flawed. Moreover, we cannot make an assumption that hardware used by a customer produces optimal performance. In order to avoid misleading predictions, we follow the below simple rules to collecting and scrubbing data.

1. Customer environments with known performance problems or issues in the recent time should not be included in the study.

2. Only dedicated environments should be collected. i.e if a customer is using the hardware for other applications, then the environment should not be considered for study.

3. At least twenty customer environments should be analyzed to feed the model. If there are outliers, data points which fall outside the cluster, then additional customer data should be collected.

4. Logs should be analyzed to find out how the application is used. Making an assumption that all licensed users are using the application can result in a flawed model. Actual usage should reflect the typical workload characteristics. Using

data from an environment that is used in a unique way can also produce skewed results.

5. Population should include environments ranging from a small to enterprise customer to cover the wide range of utilization. Sample will also contain equal number of Oracle and SQL server customers to uniformly represent both database environments.

Chapter 4 - Results

Logs from the production environments were analyzed. Application web logs have the URL (web site address) and the time of the activity. Based on the logon and logoff activity, the number of concurrent users and average duration of session is calculated. System has a default inactivity timeout of 30 minutes. Any session missing logoff URL is taken as 30 minute session. Going through the web logs to collect all this information is a very tedious process. A script is used to search through logs for specific URL and count the occurrences. Analyzing the web logs from one of the production systems for the entire duration of the pay-cycle provides the following information on the workload characteristics.

The system (application server) is heavily utilized on a Monday. The peak utilization window is around a 4-hr window. In the below graph (Figure 3), there is very high activity on 5/19, 5/26 and 6/2 for a period of 4 hours. This data helps to validate the assumption about the typical workload characteristics of the application. The typical characteristic of the application is that the heaviest activity happens during the payroll processing day. The below graph confirms the typical workload.

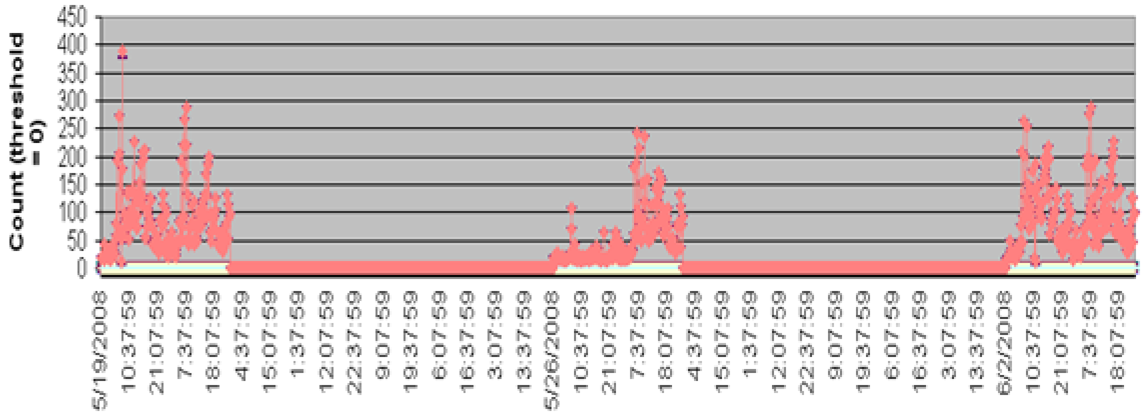


Figure 3: Application Utilization Analysis

The graph (Figure 4) below shows the maximum concurrent users on the system (application server) during the busiest period. The environment is licensed for 1800 users and about 196 concurrent users are on the server during the peak window.

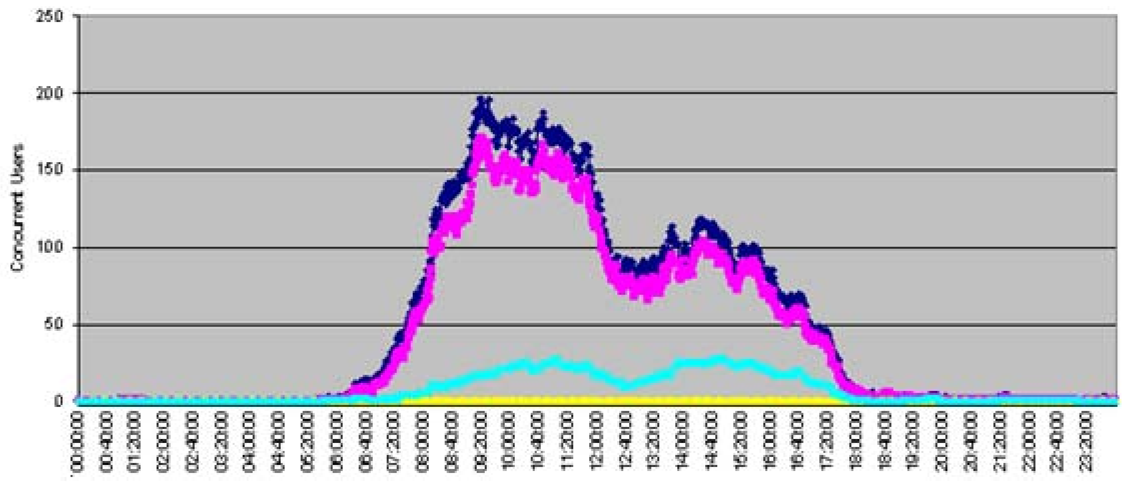


Figure 4: Concurrent Users Analysis

Given a 4 hour window and 1800 users, if we assume a uniform arrival rate, a user would arrive every eight seconds (4*60*60/1800). In any 21 minutes (average session time per user), there will be 158 (21*60/8) users on the application. We assume 100% overlap with the users who arrived in the previous 21 minutes. Therefore, the maximum concurrent users on the application are at the most double the number of the users who can arrive in a given 21 minute interval. In this environment, it is safe to expect up to 316 concurrent users. So if we know the total number of users and the average session time per user, using this calculation we can predict the approximate maximum number of concurrent sessions.

We can also calculate the number concurrent users if we assume that users arrival rate follows normal distribution. Given a four hour busy window, total number of users and the average session time, we can arrive at the number of concurrent users during peak usage.

In all the environments monitored, the number concurrent users did not exceed 1.5 times the number of users who can arrive in a window of time equivalent to the average session time. The formula to calculate the concurrent users is just an effort to make a reasonable assumption given the licenses purchased. If a large sample is used for the study, the formula for arriving at maximum concurrent users can be established with more than 95% confidence level.

In the above environment, the hardware used for the application is two servers with Intel Xeon 2 Ghz (1333 Mhz) with Dual Core. The average CPU utilization during the peak window was below 37%. We used our internal model to convert the CPU utilization into SpecInt2006 rating of 33. Our internal model computes demand with the goal of keeping the average utilization below 50%. In other words, the SpecInt rating computed reflects the CPU power needed in order to keep the average utilization around 50%.

We can also quantify the server demand by getting the SpecInt rating for the processor from Spec.org. While this may appear tricky and even extremely difficult, it is one of the reliable methods to convert different CPU models to a common scale for meaningful comparison. Our internal model uses the SpecInt2006 ratings published. Whatever is the method used, we just need to be consistent in applying that to all the observations.

Here is another observation from a production environment. The number of concurrent users during the peak activity is 18. The peak duration is a six hour window from 6 a.m. to 12 p.m. The total number of licensed users is 100. The average session time is 40 minutes.

If we assume uniform arrival within the six hour window, users arrive at the rate of one for every 216 seconds (6*60*60/100). In any 40 minute (average session time) period, we can expect 40*60/216 (12) users to arrive. With 100% overlap with the previous 40 minutes, we can expect at the most 24 concurrent users. The application server hardware used is a Dual CPU 2 Ghz (400) with the SpecInt 2006 rating of 3.97.

Logs from different production environments were analyzed. The following statistics are gathered from each environment.

1. Peak Duration Window – Duration of time when the application was highly active.
2. Number of actual concurrent users during the peak utilization
3. Average Session time
4. Reports run per user
5. CPU utilization on the application and the database server
6. Hardware used converted to SpecInt2006 rating.
7. Number of Licensed Employees
8. Number of Licensed Supervisors
9. Database Platform
10. Modules implemented

Below (Table 2) is the summary of statistics collected from 12 different production environments.

Well known independent variables						Independent Variables Derived from Empirical Data				Observed Dependent Variables	
		Optional Modules			Database SQL(0)		In Minutes		In Hours		
Licensed Employees	Licensed Managers	Leave	Attendance	Scheduling	Oracle(1)	Concurrent Users	Av. Session Time	#Reports/User	Peak Duration	App Server Rating	Database Server Rating
8000	1800	0	0	0	1	196	21	2	4	33.66	11.45
17500	800	0	0	0	1	137	26	1	4	24.33	5.66
12000	1200	0	1	1	1	221	35	5	5	49.02	15.16
8500	850	1	1	0	0	96	17	1	4	20.65	10.48
8000	2500	1	0	0	1	331	20	1	4	82.45	14.62
1900	350	0	1	0	0	42	18	1	4	7.63	4.6
3500	125	0	0	1	0	20	24	2	4	4.34	2.74
24610	100	0	0	0	0	18	40	3	6	3.97	2.38
9100	700	0	1	0	1	205	45	4	4	43.47	17.14
25000	1500	0	0	0	1	286	29	5	4	48.85	13.01
6500	500	0	0	0	0	63	19	2	4	10.27	6.96
16000	1200	0	1	0	1	122	23	5	6	22.41	10.01

Table 2: Data collected

All the independent variables listed above affect the utilization of the application and the database hardware. We are looking for a relationship between the independent variables and the CPU demand so that the computing resource on the application and the database server can be predicted. Benchmark testing has shown that performance varies linearly with workload up to a certain limit (100,000 licenses). For any application, this is true. Beyond a particular point, adding more resources does not improve the performance. Since the relationship is linear in nature and there is a strong correlation between the user activity and the resources deployed, multiple regression technique is used in this study. We use forward-selection stepwise regression to select independent variables one by one. The first variable included is the one that has the highest R square value for predicting the application server demand. The second variable included is the one that, when combined with the first one, produces the highest adjusted R square value. The third variable included yields the highest adjusted R square value. This process is continued to include all variables that increases the probability of predicting the server demand. We use adjusted R square to see if the added independent variable improved the predictability of the model.

The graph (Figure 5) below shows the linear relationship between the application computing resource and the concurrent users.

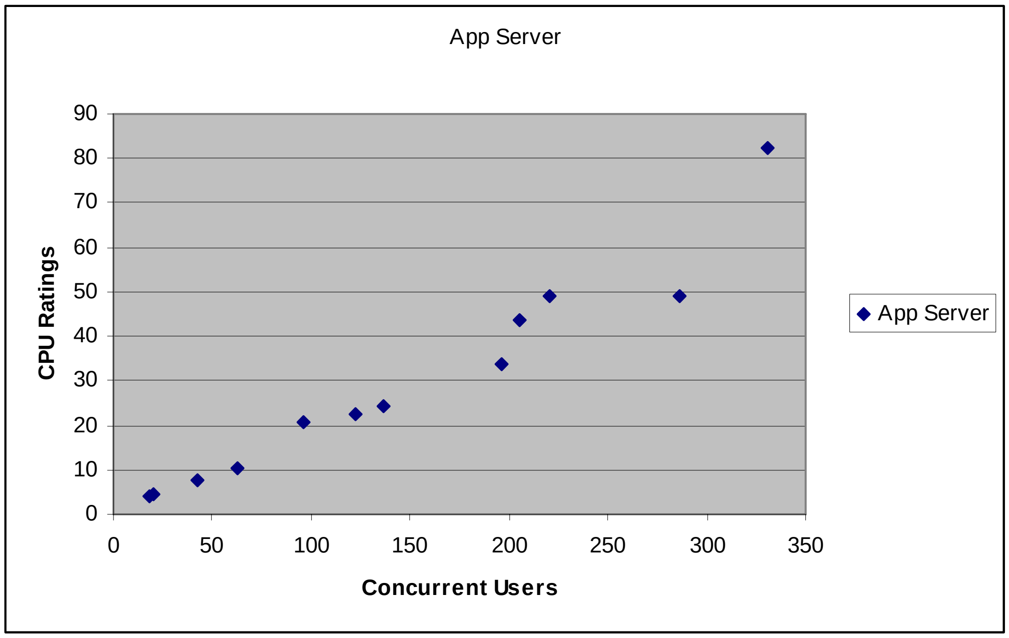


Figure 5: Users/CPU relationship

Below (Table 3) is the R Square value from the single regression between the application server demand (CPU rating) and the individual variable.

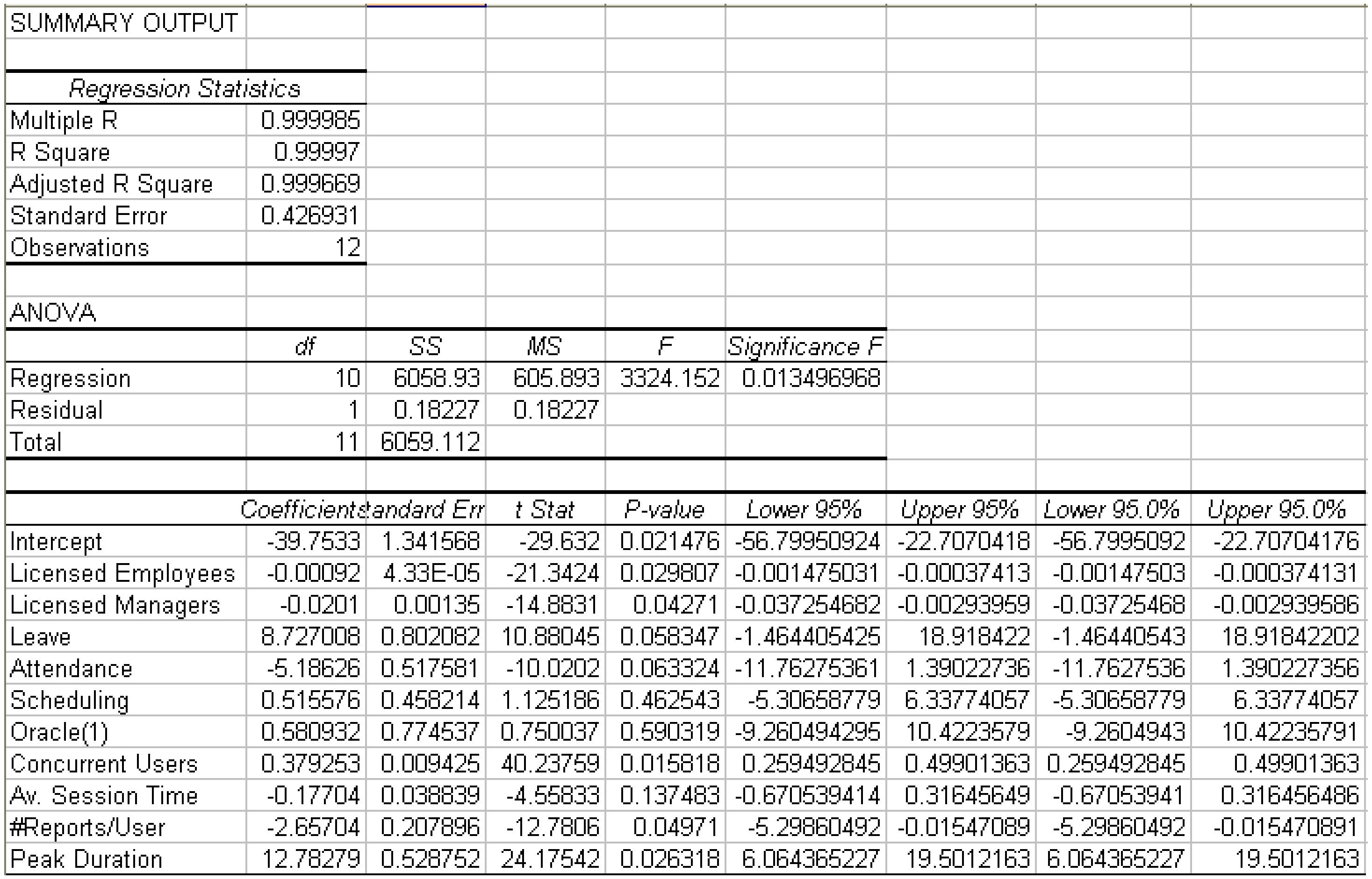
Variables	R Square
Concurrent Users	0.94
Managers	0.77
Database	0.56
Leave	0.2
Peak Duration	0.05
#Reports	0.04
Average Session Time	0.01
Employees	0.0079
Schedule	0.003
Attendance	0.0005

Table 3: Application Server CPU and Variables Correlation

The above table (Table 3) clearly shows that there is a very high correlation between the number of concurrent users and the CPU resource used on the application server. Multiple Regression is performed by adding the next independent variable (Mangers) to the Concurrent Users. This process is repeated and the R Square (Coefficient of Determination) and Adjusted R Square values were recorded.

The highest coefficient of determination and adjusted R Square was produced when all the variables were included in the model. In regression analysis, R Square is a good measure of relationship between the independent and the dependent variable. But when there are several independent variables and the analysis is performed on a sample of data, R Square Adjusted is a much more realistic measure of the correlation.

Below (Table 4) is the output of the multiple regression using all independent variables for predicting the application server demand. The model has very high R Square and Adjusted R Square, which leads to the conclusion that variables included have very high predictability of the dependent variable. Also the Significance F value less than 0.05 indicates that the regression is significant at 95% confidence level. The F test merely indicates that there is a good evidence of some degree of predictive value somewhere among the independent variables (Hildebrand and Ott, 1998). It does not give any direct indication of how strong the relation is, or any indication of which individual independent variables are useful.



SUMMARY OUTPUT

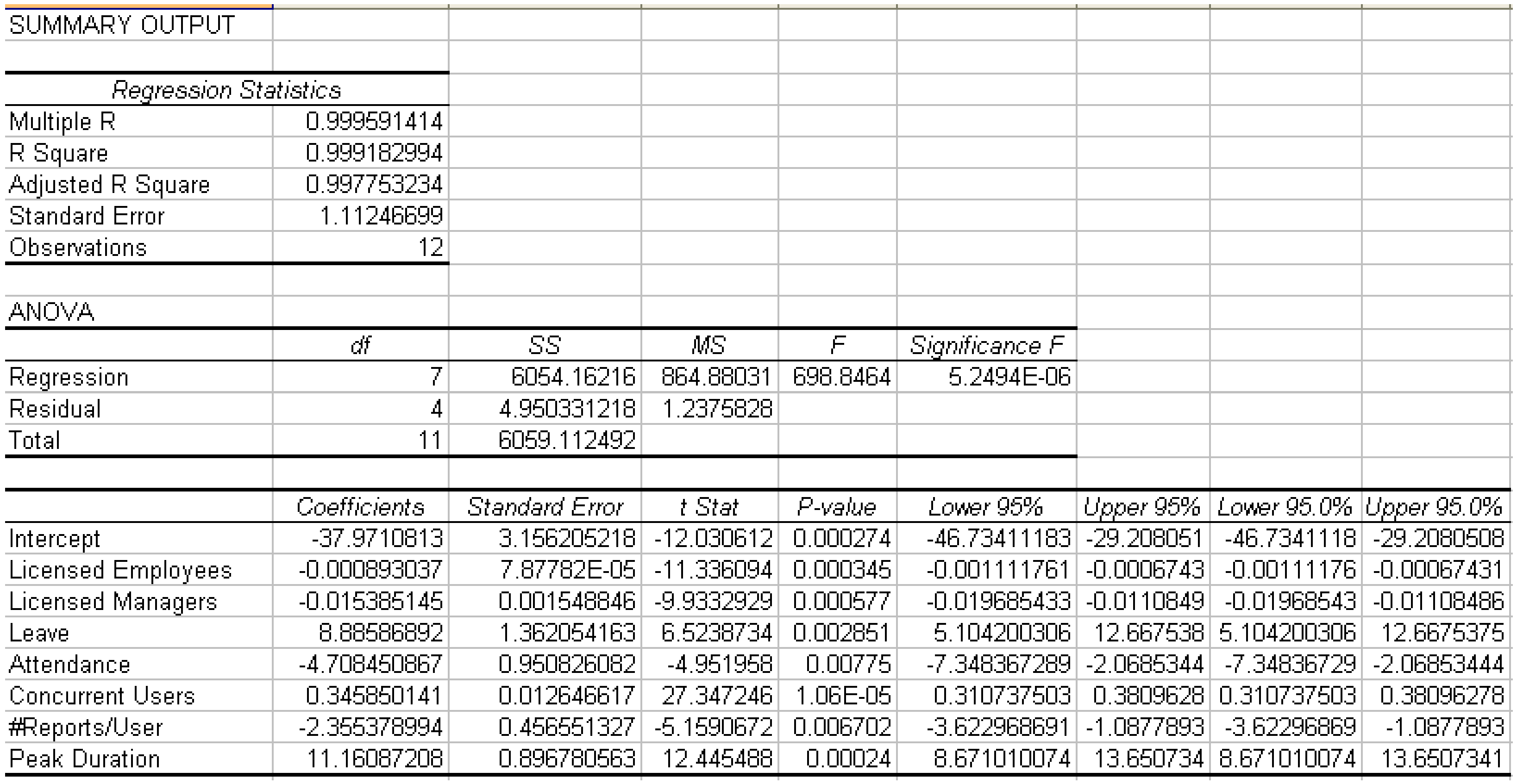
Regression Statistics	
Multiple R	0.999985
R Square	0.99997
Adjusted R Square	0.999669
Standard Error	0.426931
Observations	12

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	10	6058.93	605.893	3324.152	0.013496968
Residual	1	0.18227	0.18227		
Total	11	6059.112			

	Coefficients	*Standard Err*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-39.7533	1.341568	-29.632	0.021476	-56.79950924	-22.7070418	-56.7995092	-22.70704176
Licensed Employees	-0.00092	4.33E-05	-21.3424	0.029807	-0.001475031	-0.00037413	-0.00147503	-0.000374131
Licensed Managers	-0.0201	0.00135	-14.8831	0.04271	-0.037254682	-0.00293959	-0.03725468	-0.002939586
Leave	8.727008	0.802082	10.88045	0.058347	-1.464405425	18.918422	-1.46440543	18.91842202
Attendance	-5.18626	0.517581	-10.0202	0.063324	-11.76275361	1.39022736	-11.7627536	1.390227356
Scheduling	0.515576	0.458214	1.125186	0.462543	-5.30658779	6.33774057	-5.30658779	6.33774057
Oracle(1)	0.580932	0.774537	0.750037	0.590319	-9.260494295	10.4223579	-9.2604943	10.42235791
Concurrent Users	0.379253	0.009425	40.23759	0.015818	0.259492845	0.49901363	0.259492845	0.49901363
Av. Session Time	-0.17704	0.038839	-4.55833	0.137483	-0.670539414	0.31645649	-0.67053941	0.316456486
#Reports/User	-2.65704	0.207896	-12.7806	0.04971	-5.29860492	-0.01547089	-5.29860492	-0.015470891
Peak Duration	12.78279	0.528752	24.17542	0.026318	6.064365227	19.5012163	6.064365227	19.5012163

Table 4: Application Server Regression Output



SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.999591414
R Square	0.999182994
Adjusted R Square	0.997753234
Standard Error	1.11246699
Observations	12

ANOVA

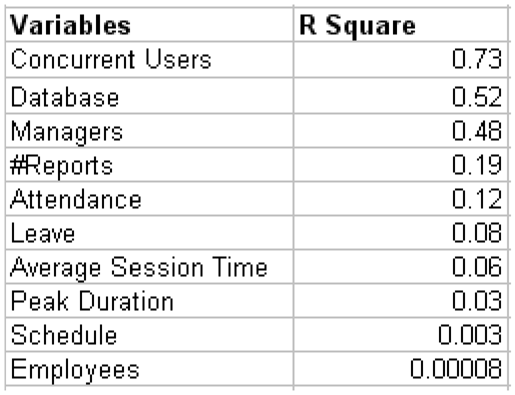
	df	*SS*	*MS*	*F*	*Significance F*
Regression	7	6054.16216	864.88031	698.8464	5.2494E-06
Residual	4	4.950331218	1.2375828		
Total	11	6059.112492			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-37.9710813	3.156205218	-12.030612	0.000274	-46.73411183	-29.208051	-46.7341118	-29.2080508
Licensed Employees	-0.000893037	7.87782E-05	-11.336094	0.000345	-0.001111761	-0.0006743	-0.00111176	-0.00067431
Licensed Managers	-0.015385145	0.001548846	-9.9332929	0.000577	-0.019685433	-0.0110849	-0.01968543	-0.01108486
Leave	8.88586892	1.362054163	6.5238734	0.002851	5.104200306	12.667538	5.104200306	12.6675375
Attendance	-4.708450867	0.950826082	-4.951958	0.00775	-7.348367289	-2.0685344	-7.34836729	-2.06853444
Concurrent Users	0.345850141	0.012646617	27.347246	1.06E-05	0.310737503	0.3809628	0.310737503	0.38096278
#Reports/User	-2.355378994	0.456551327	-5.1590672	0.006702	-3.622968691	-1.0877893	-3.62296869	-1.0877893
Peak Duration	11.16087208	0.896780563	12.445488	0.00024	8.671010074	13.650734	8.671010074	13.6507341

Table 5: Application Server Regression Output 2

In the output shown in Table 4, there are several independent variables with the p-value greater than 0.05 or even 0.1. A p-value is a measure of how much evidence we have against the null hypothesis that the individual independent variable has no additional predictive value over and above that contributed by the other independent variables. In Table 4, the independent variables ‘Scheduling’, ‘Oracle’, and ‘Av. Session Time’ have higher P-value (greater than 0.1). Adding these variables after including all other independent variables would not improve the prediction. Table 5 is the output of regression after removing the statistically insignificant variables.

Below (Table 6) is the R Square value from the single regression between the Database Server demand (CPU rating) and the individual variable.



Variables	R Square
Concurrent Users	0.73
Database	0.52
Managers	0.48
#Reports	0.19
Attendance	0.12
Leave	0.08
Average Session Time	0.06
Peak Duration	0.03
Schedule	0.003
Employees	0.00008

Table 6: Database CPU and Variables Correlation

Table 7 shows the output of multiple regression for predicting the database server utilization. Based on the sample collected, it is hard to reject the null hypothesis that independent variables have no predictive power. Significance F and the P-values are higher than 0.05 or even 0.1. The width of the interval for the variables is also large. All the intervals (Lower & Upper 95%) include 0.

SUMMARY OUTPUT

Regression Statistics	
Multiple R	0.985672
R Square	0.971549
Adjusted R Square	0.687041
Standard Error	2.799975
Observations	12

ANOVA

	df	*SS*	*MS*	*F*	*Significance F*
Regression	10	267.7183662	26.77184	3.414837	0.39974685
Residual	1	7.839858833	7.839859		
Total	11	275.558225			

	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	7.319375	8.798510354	0.831888	0.558259	-104.476299	119.115049	-104.476299	119.115049
Licensed Employees	-0.00015	0.000284118	-0.51573	0.696874	-0.00375659	0.00346353	-0.00375659	0.003463533
Licensed Managers	0.004681	0.008855966	0.528614	0.690427	-0.10784433	0.1172071	-0.10784433	0.117207098
Leave	3.88948	5.260356074	0.739395	0.594677	-62.9496809	70.7286416	-62.9496809	70.72864159
Attendance	2.148438	3.394492911	0.632919	0.640772	-40.9826839	45.2795599	-40.9826839	45.27955989
Scheduling	-1.66809	3.005143107	-0.55508	0.677404	-39.8520494	36.5158778	-39.8520494	36.51587775
Oracle(1)	1.015205	5.079707976	0.199855	0.874423	-63.528605	65.5590141	-63.528605	65.5590141
Concurrent Users	-0.00998	0.061815007	-0.16144	0.898104	-0.79541345	0.77545482	-0.79541345	0.775454817
Av. Session Time	0.240647	0.25472177	0.944745	0.518083	-2.99589976	3.47719414	-2.99589976	3.477194144
#Reports/User	1.767716	1.363459817	1.296493	0.41826	-15.5566837	19.0921155	-15.5566837	19.09211548
Peak Duration	-2.74056	3.467753497	-0.7903	0.57423	-46.8025489	41.3214229	-46.8025489	41.32142286

Table 7: Database Regression Output

Chapter 5 – Key Findings

At first glance, the model results may not seem like a discovery of any significance, but looking at the equation closer, we can see the wide applicability of this modeling technique.

For predicting the application server demand, an equation like below can be used –

= (8.89*Leave + 0.346*Concurrent Users + 11.16*Peak Duration) - (37.97 + 0.015*Licensed Managers + 4.7*Attendance + 2.36*Reports/User)

The values are based on the information gathered from real life production environments, an extremely realistic and better than in-house benchmark values. Gathering multiple data points from a real life production environment will provide all information we need to derive the unknowns. By understanding the production workload and carefully analyzing the logs, we can arrive at reasonable assumptions on the unknowns. In the above equations, the only unknowns are -

- number of concurrent users
- number of reports per user
- duration of the peak window

We already know how to compute the approximate number of concurrent users using the formula based on the average session time and the duration of peak window and the total number of users.

Given a normal workload characteristic, we know the value for peak duration window is in the range of four to six hours, the value for average session time is in the range of 17 to 45 minutes and the value for reports per user is in the range of one to five. Depending on whether we want conservative, average or aggressive estimate, we can use the minimum, average or the highest values for these variables in the above equations.

A simple spreadsheet calculator can be built which will take all the known variables and the type of estimate (conservative, average or aggressive) to produce the demand for the application and the database server.

Chapter 6 -- Suggestions for Additional Work

The samples collected in this study did not target any particular industry or market. Also, the samples collected included a wide range of products and users. Although the model uses real life data, it has some known defects.

1. The number of samples used in the study is less than 20. We need at least 20 to 30 samples to establish a credible model.

2. There are three variables whose values have to be assumed based on the range of data collected.

3. The database utilization could not be predicted with higher confidence level based on the sample data collected.

Model can be improved by collecting 20 samples from each industry vertical (like education, government, manufacturing, healthcare etc). Within each vertical industry, we need to collect samples based on customer size (1000, 5000, 10000 user licenses). This will really help in understanding the workload characteristics within each industry and customer size (small, mid-market, enterprise etc). A more realistic assumption can be made on the unknown variables. However, a study like this would require considerable effort and time.

Another way to simplify sizing is to collect enough samples (20-30) in each range of customer size. For example, collect 20 samples of hardware from customers with less than 1000 employees. The maximum CPU rating in each category (like 1000, 3000, 5000 users etc) will serve as the hardware recommendation that will be used for the hardware planning during the sales cycle for a new customer in the same category. This method will not only be quick but will also avoid any kind calculation. Moreover, because we recommend the highest hardware resource used in each category, we can be assured that there is enough room for growth or scalability in the solution.

The model established in this study is not conclusive. But it helps to provide enough information to begin exploring the possibilities, usefulness and appropriateness of theoretical models based on empirical or production data to arrive at quick sizing. Models to predict CPU, Disk Space, I/O throughput, Memory requirements can be created by carefully collecting metrics from production environments. Once the metrics are defined and carefully collected, a simple estimation technique like regression analysis can be performed to build a fairly high confidence model.

The log analysis technique (to collect the independent variables) used in this paper can be applied in other scenarios (like performance troubleshooting) to validate the assumptions made during sizing and also to understand the application usage characteristics.

References

Benton Gibbs G., Jerry M. Enriquez, and Nigel Griffiths, eds. (March 2004). IBM @server pSerries Sizing and Capacity Planning (Ibm.com/Redbooks). http://www.redbooks.ibm.com/redbooks/pdfs/sg247071.pdf (accessed April 1 2009)

Chatterjee, Samprit and Hadi S. Ali. 2006. *Regression Analysis by Example*. San Francisco: John Wiley & Sons.

Cook, David R., Ellen M. Dudar, and Shallahmer A. Craig. (1997). The Ratio Modelling Technique. http://www.geocities.com/mtarrani/CapacityRatioModeling.pdf (accessed March 20, 2009)

Microsoft Corporation (White Paper). (June 2003). Windows Sever 2003 Terminal Server Capacity and Scaling. http://www.microsoft.com/windowsserver2003/techinfo/overview/tsscaling.mspx (accessed April 2, 2009)

Hildebrand, David H., Lyman R. Ott. 1998. *Statistical Thinking for Managers*. Belmont: Duxbury Press

Pedigo, Larry. (2004). "Sizing Oracle on Microsoft Windows and Dell PowerEdge Servers (White Paper sponsored by Microsoft, Dell and Oracle)". http://www.dell.com/downloads/global/solutions/Oracle%20on%20Windows%20Sizing.pdf (accessed April 5, 2009)

Pepper, Jason, Jack Sun, and Biswajit Nayak. (2004). How to Effectively Size Hardware for your Portal implementation. http://www.oracle.com/technology/products/ias/portal/pdf/oow_10gr2_1337_pepper.pdf (accessed March 30, 2009)

Shallahamer, Craig A. (2002). Predicting Computing System Throughput and Capacity. http://www.geocities.com/mtarrani/PredictingComputingCapacity.pdf (accessed March 28, 2009)

Sun Microsystems, Inc (White Paper). (2002). Sun Server Scalability and Sizing Guide. http://www.sun.com/servers/white-papers/scalability-sizing-guide.pdf (accessed March 31, 2009)

Appendix A - Criteria for Hardware Sizing

Criteria Factor	Value
The peak window duration (in hours) is:	4
Editing is done by:	Administrators
The total number of Users	750
The Average session time (in minutes) is:	19
The number of Users accessing the HTML interface	750
Will Scheduling be used?	Yes
Accruals Used	No
The number of report run per User:	1
The percent of Employees paid from schedule is:	0%
The percent of daily Edits	50%
The percent of heavy users is:	1%
The percent of medium users is:	6%
The percent of light users is:	93%
The total number of Employees that punch **IN** during the peak Window	4,700
The total number of Employees that punch **OUT** during the peak Window	1,600
The Maximum number of Managers that will create schedules	60
The number of Employees scheduled by these Supervisors is:	5,000
The peak window for Supervisors using Schedule:	2
The average session time for Schedulers	30
The total number of Employees that will recording time is:	10000
The number of times a day a typical employee punches is:	2
The number of days during a Pay Period a typical Employee works is	10
The average number of Transfers per Employee per day is:	1
The number of Manual edits typically performed to an Employee per Pay Period is:	6
The percentage of Supervisors that will make edits to their Employees on a daily basis is:	50
The percentage of Employees that require manual edits during a Supervisor edit session is:	50
The percent of Labor Account Transfers that will require an edit by a Supervisor during a Pay Period is:	7
Number of Collection Devices	50
Number of Workflow Processes	5

Appendix B - Performance Terminology

Concurrency
The ability to handle multiple requests or users simultaneously. Threads, processes are examples of concurrency mechanism.

Contention
Competition for resources on the servers hosting the application and the database

Cluster
A group of machines that handle workload in a distributed manner, providing redundancy and failover

Failover
A method of allowing one machine or set of machines to provide an alternative execution arena for a task, should the original machine(s) fail.

Hit
The subsequent request for a snippet of content from the web application

Latency
The time that one system component spends waiting for another component in order to complete the entire task. Latency can be defined as wasted time. In networking contexts, latency is defined as the travel time of a packet from source to destination.

Page request
The unique request for a page defined inside the application. A figure specifying page requests per second is the measurement of the load expected for the architected solution given a common element of web content.

Response time
The time between the submission of a request and the receipt of the response

Scalability
The ability of a system to provide throughput in proportion to, and limited only by, available hardware resources. A scalable system is one that can handle increasing numbers of requests without adversely affecting response time and throughput. A system exhibits good scalability when the amount of system resources consumed increases at the same rate as the load is increased without adversely impacting response times or throughput.

Service time

The time between the receipt of a request and the completion of the response to the request

Think time
The time the user is not engaged in actual use of the processor.

Stream time
The time taken to transmit the response to the requestor

Throughput
The number of requests processed per unit of time.

Wait time
The time between the submission of the request and initiation of the request