TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Jari Soininen

Website Performance Evaluation and Estimation in an E-business Environment

Julkaisu 1083 • Publication 1083

Tampere 2012

Tampereen teknillinen yliopisto. Julkaisu 1083
Tampere University of Technology. Publication 1083

Jari Soininen

Website Performance Evaluation and Estimation in an E-business Environment

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Auditorium 240, at Tampere University of Technology – Pori, on the 31st of October 2012, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2012

ISBN 978-952-15-2929-0 (printed)
ISBN 978-952-15-2957-3 (PDF)
ISSN 1459-2045

Soininen, Jari, 2012, "Website Performance Evaluation and Estimation in an E-business Environment". Faculty of Computing and Electrical Engineering. Tampere University of Technology, Tampere, Finland.

Keywords: Performance Engineering, E-commerce, E-business, Optimisation, Performance Monitoring, Natural Load

Abstract

This thesis introduces a new *Predictus*-model for performance evaluation and estimation in a multi-layer website environment. The model is based on soft computing ideas, i.e. simulation and statistical analysis. The aim is to improve energy consumption of the website's hardware and investment efficiency and to avoid loss of availability. The aim of optimised exploitation is reduced energy and maintenance costs on the one hand and increased end-user satisfaction due to robust and stable web services on the other.

A method based on simulation of user requests is described. Instead of ordinary static parameter set, the dynamic extraction from previous log files is used. The distribution of existing requests is exploited to generate the actual based natural load. By loading the server system with valid and well-known requests, the behaviour of the server system is natural. The control back loop on the generation of work load assures the validity of the work load in the long-term.

A method for identifying the actual performance of the website is described. Using the well-known load in simulation of usage by a large number of virtual users and observing the utilisation rate of server resources ensure the best information for the internal state of the system. The disturbance of the service website usage can be avoided using the mathematical extrapolation method to reach the saturation point on the single server resource.

Preface

The present study was carried out during the period 2006-2012 at the Josbit Ltd and Tampere University of Technology, Pori. The realization that performance measurement requires tremendous manual labour was initially formed in the mid-1990s. However, it took more than ten years before I discovered that this purpose can be developed into a systematic model which can be further automated through programmatic means.

I wish to express my gratitude to Mr. *Teijo Lallukka* and Mr. *Timo Karvonen* for their technical assistance. Without their help, the results presented in this thesis would not have been achieved.

I owe my utmost thanks to my colleague, Mr. *Aki Parviainen*, for his close cooperation and many helpful discussions during all these years. His expertise in e-commerce has been very useful.

A vote of thanks also to Maggy O'Donnell for her assistance with the language. Thanks to her skills, this thesis is now easier to read.

I wish to express my sincere gratitude to my advisers, Prof. *Hannu Jaakkola* and Dr. *Jari Palomäki*, from Tampere University of Technology for their invaluable encouragement, guidance and forbearance with my thesis. I want to thank Ms. *Ulla Nevanranta* for taking care of all the practical arrangements during the research. Sincere acknowledgements go also to Prof. Dr. rer. nat. habil. *Bernhard Thalheim* and D.Sc., Leading Researcher *Enn Tõugu* for reviewing the thesis and for providing valuable comments on the manuscript.

The work has been supported financially by Academy of Finland, Ulla Tuominen Foundation, High Technology Foundation of Satakunta, and Finnish Cultural Foundation, which greatly contributed to the preparation of this thesis and for which I want to express my highest appreciation.

Finally, my warmest thanks go to my dearest soul mate Soile, and also to my wonderful children Joona and Jemina, whom lately I have not been able to spend as much time with as a father should.

Table of Contents

Table of Contents

List of Figures

List of Figures

List of Figures

List of Tables

Acronyms and Initialisms

ACID	Atomicity, Consistency, Isolation, and Durability
ATM	Asynchronous Transfer Mode
ANOVA	ANalysis Of Variance
ASMP	ASymmetric MultiProcessor
CDF	Cumulative Distribution Function
CMG	Computer Measurement Group
CPU	Central Processing Unit
DBMS	DataBase Management System
DNS	Domain Name Server
FCFS	First Come, First Served
FIFO	Fist In, First Out
FILO	Fist In, Last Out
HPE	Human Performance Engineering
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
I/O	Input/Output
IID	Independent and Identically Distributed
IIS	Internet Information Server
IP	Internet Protocol
IT	Information Technology
ITIL	Information Technology Infrastructure Library
LAN	Local Area Network
LAN	Local Area Network
LCG	Linear Congruential Generator
LFU	Least Frequently Used
LIFO	Last In, First Out
LRU	Least Recently Used

MIMD	Multiple Instruction, Multiple Data
MISD	Multiple Instruction, Single Data
MPMD	Multiple Programme, Multiple Data
MPP	Massively Parallel Processor
MRU	Most Recently Used
MTBF	Mean Time Between Failures
MTTF	Mean Time To Failure
MTTR	Mean Time To Repair
MVA	Mean Value Analysis
NAS	Network Attached Storage
NUMA	Non-Uniform Memory Architecture
OLAP	On-Line Analytical Processing
OLTP	On-Line Transaction Processing
OS	Operating System
PDF	Probability Density Function
PMF	Probability Mass Function
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RDBMS	Relational DataBase Management System
RFC	Request For Comments
ROT	Rule Of Thumb
SAN	Storage Area Network
SIMD	Single Instruction, Multiple Data
SISD	Single Instruction, Single Data
SLA	Service Level Agreement
SMP	Symmetric Multi Processor
SPE	Software Performance Engineering
SPEC	Standard Performance Evaluation Corporation
SPMD	Single Program, Multiple Data
TCP	Transport Control Protocol
TPC	Transaction Processing Council

UDP	User Datagram Protocol
UMA	Uniform Memory Architecture
URL	Universal Resource Locator
VLAN	Virtual Local Area Network
VM	Virtual Memory
WAN	Wide Area Network

1 Introduction

This chapter provides a short overview of the research topic examined in this thesis. The aim of this chapter is to provide the background to the research. Such a research background includes an introduction to the research area and the motivation behind this research.

1.1 Background

The World-Wide-Web was initially started as a project to enable the easy exchange of information between researchers who were geographically distant from each other (Berners-Lee, Caillan, Luotonen, Nielesen, & Secret, 1994). It has now taken the role of an international information superhighway, and it has become synonymous with a mega warehouse of information. Web services have been used for different purposes, such as allowing for the exchange of information within and between organizations, and lately for advertising, selling and buying of merchandise, which have been referred to as electronic commerce, normally known as e-commerce. The World-Wide-Web has certainly helped in the sharing of information among Internet users throughout the world. There can be no denying that the Web has already become a part of everyday life in a large part of the world.

Many traditional services have been transformed or converted to Web-based services. A variety of e-commerce online models now exist, ranging from shopping, auction, reservation, media services, banking and trading to customer relation management, personnel management, etc. In addition to becoming a cost-effective solution for many traditional businesses, e-commerce is also creating new business opportunities. Web-based services have become such a critical component of many companies nowadays that guaranteeing performance and availability has become essential. About a third of all companies enabled customers to order their products or services online in 2007 (Selhofer, Lilischkis, Woerndl, Alkas, & O'Donnell, 2008). This can be accomplished through different technical channels, including the company's own website or third party trading platforms on the internet.

1 Introduction

Splaine & Jaskiel (2001) pointed out two relevant questions in system management: “How much extra performance do we need to have at present?” and “How long before we have to upgrade the existing infrastructure?” Maintaining the web system infrastructure should meet a two-fold challenge (Lin, Liu, Xia, & Zhang, 2005). It must meet customer expectations in terms of quality of service (QoS) and that companies have to control information technology (IT) costs to remain competitive. In addition, the aim in most cases of e-commerce services is to keep the service attractive to users and possibly increase the number of interested visitors. To achieve this requirement, services have to be constantly updated and improved with new service components (applets) to keep them coming. The improvements and enhancements of services require a more effective infrastructure to guarantee adequate quality of service. In many cases, customer expectations are satisfied by increasing the performance of service systems through adding more resources such as processors, memory, and even more redundant servers.

ITIL (APM Group Ltd, 2011) defines the term *performance* as *a measure of what is achieved or delivered by a System, person, team, Process or IT Service*. In addition, in this study performance is embodied as an index. It is understood as a score for how well a particular workload's set of assigned resources are being utilized compared to the optimum level. This index immediately shows whether resources have remaining capacity, are being over-utilized, or are aligned "just right" to meet the demand.

Performance requirements are based on the requirements during *hotspots*. The term ‘hotspot’ is introduced in (Baryshnikov et al., 2005) and it means anything where traffic is significantly higher than the norm. The performance requirements vary depending on a number of reasons like the time of the day, happenings in society, marketing activities, etc. This could easily lead to overestimation of performance requirements and to increased information technology costs. To avoid overloading of the service, the gap between actual usage of the service and total available performance should be known by the service provider. An accurate knowledge about the web server system bottlenecks and their locations is not enough in many cases. On those cases, the improvements of system activities are

focused via rough guesses. Some of those improvements might be focused correctly. However, some improvements are merely extra costs. To avoid unnecessary high performance and high costs, it can be done by optimising a web service regularly.

Despite the increasing urgency of getting a hold on soaring energy demands in IT, costs continue to spiral out of control. Energy costs are increasing by 16% every year, while greenhouse gas emissions from data centres have already surpassed the output of Argentina or Netherlands and are due to overtake those of all airlines by 2020 (Kaplan, Forrest, & Kindler, 2008). However, according to one report, it need not be that way. The report recommended that companies adopt a new metric called Corporate Average Data Efficiency (CADE) which combines both IT and facilities' costs to monitor energy use, and create "energy czar" positions to manage energy efficiency.

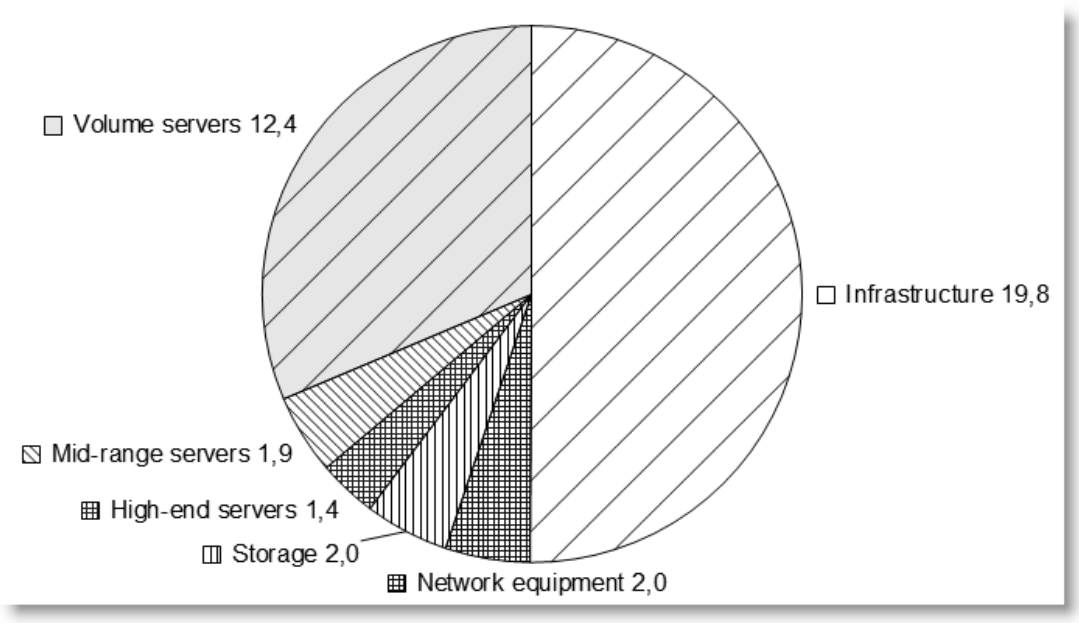

Figure 1.1: Electric power consumption in data centres in EU in 2006. The chart shows the energy consumption (in TWh) of server hardware (volume, mid-range, high-end), storage and network equipment and infrastructure (cooling, UPS, lighting etc.) of total power consumption.

Power consumption in data centres in the U.S. has been studied by Koomey (2007; 2011). In 2005, the total direct consumption for all servers in the U.S. was about 5 million kW, including cooling and auxiliary equipment. When electricity used for cooling and auxiliary equipment is included, it rises to 1.2% of retail electricity

sales in that year, resulting in a utility bill of 2.7 billion USD (the 2006 dollar value) when valued at U.S. industrial electricity prices. The total server power and electricity consumption for the world as a whole is about two and a half times bigger than that of the U.S. The E-Server Consortium has reported corresponding energy consumption in Europe (Schäppi, Bellosa, Przywara, Bogner, & Weeren, 2007). The report shows that the total electric power consumption in Western Europe (EU 15 plus Switzerland) in 2006 amounted to 39.5 TWh. In Figure 1.1, it is shown that the consumption is concentrated mostly on infrastructure and on the class of low price servers. According to the report, the servers are classified into three categories, the volume servers (valued <25,000 US$), mid-range servers (25,000–500,000 US$), and high end servers (>5,000,000 US$). Data centres' energy consumption growth rate appears to have slowed significantly in the period 2005-2010. The projected growth rate was 100% (Koomey, 2007), but what was achieved seems to have been 36% (Koomey, 2011). This can be explained, above all, by the economic slowdown rather than as a result of active energy-saving measures.

Updated information within the EU, or global data in general, does not exist, but according to Figure 1.2 (Koomey, 2011), distribution of the different categories of servers would have remained the same for the U.S. data centres between 2005 and 2010.

Server utilization data centre is, in one of the reports (Kaplan et al., 2008), only 6%. According to a second source of utilization, it is about 15% (VMware, n.d.). The centralization of services in a smaller number of servers can be the average rate of utilization to increase.

In may ways, Green ICT has risen in the recent past according to the discussion. It will be completed under the heading of constantly developing new scientific research (Benedetto et al., 2012; Hu, Deng, & Wu, 2011; Jiang, 2012; Sugiyama, 2012) and commercial reports (Datacenter Dynamics, 2011). In addition to these various alliances (Commission of the European Communities, 2008; GreenICT.com.au, 2008; GreenICT.org.uk, n.d.), some have created different goals for the achievement.

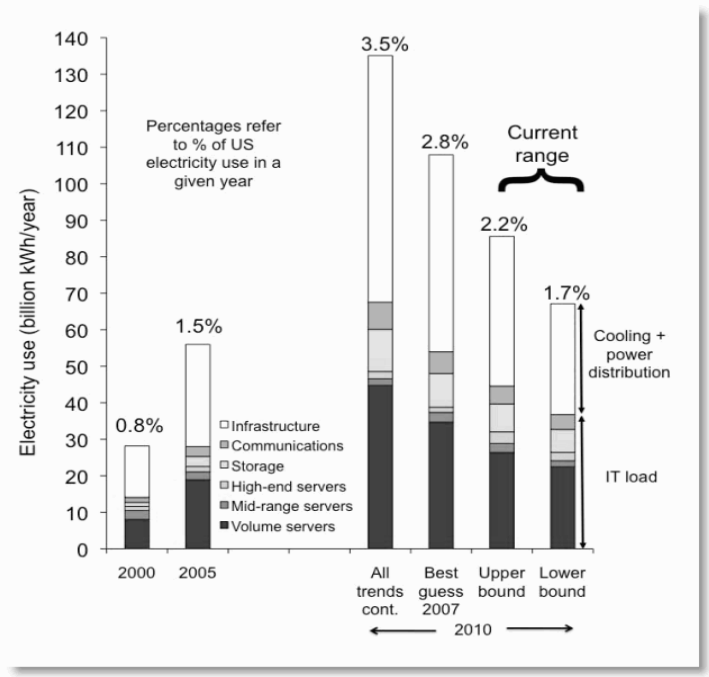

Figure 1.2: U.S. electricity use for data centres (2000, 2005, and 2010)

By combining the functions of servers, it can be used to reduce the number of server machines. IDC Report (IDC, 2007) predicts that with the x86 processors, the supply will increase by 61%. The 2010 forecast has reduced the number of deliveries to 39% per annum due to the multi-core technology and server consolidation. By reducing the number of servers, data centers are expected to decline in total consumption, which includes the purchase of new equipment, items of expenditure arising from the direct energy consumption and cooling.

The number of users of websites is generally very difficult to determine. Some figures, whose accuracy is laborious to verify, can be obtained; for example, (Miniwatts Marketing Group, 2012). It shows that at the end of 2011, the world had 2.3 billion Internet users. The total transport volume is even more difficult to assess. In any case, clearly there is an enormous number of online services, including users and traffic volume. In addition, we know that websites, applications and technology are very heterogeneous.

The performance requirements vary depending on a number of reasons, like the time of the day, events in society, marketing activities, etc. This easily leads to the overestimation of performance requirements and to increased information

technology costs. With real and sufficient information, usage of the web service will only be achieved by analyzing the log data on a regular basis. When the number of servers is reduced and the physical machine utilization rate increases, less energy is needed in data centers. In many cases, the computing power is sized on a peak load basis, and even if the devices are under-loaded, fewer machines at a higher rate of use can achieve the same result as before.

Ferrari et al. (2006) said that providers of e-business services, such as on-line banking, auctioning and retailing, must utilize their computing resources efficiently and offer a high quality of service to their users. In order to do so, it is important to predict the performance that the system can achieve for a given level of demand, for instance, using a model which is: (a) sufficiently detailed to take into account the essential system features; and (b) sufficiently simple to be analytically and numerically tractable.

1.2 Research Problem

Regular evaluation is needed to meet the challenge of quality of service and reasonable level in costs of information technology's infrastructure. There are several models in evaluating computer systems. Quite a number of them are analytical (Dilley, Friedrich, Jin, & Rolia, 1998; Jian, 1991) or simulation models (Jian, 1991). However, redundancy in high availability and scalable web server systems makes it challenging to perform a regular evaluation of performance by simulation or analytic models. The latent errors in software or configurations can slowly deprecate the effective performance of the system, making them noticeable only in long periods. This type of errors is not manageable via short period simulation or by analytical methods. In addition, changes in software or hardware configurations and new versions of applications require manual update of the analytic models. The current best practice in the area is based on experience and the intuition of web service maintainers.

Proactive management of resources requires accurate prediction of workload. A study (Sang & Li, 2000) of network traffic assesses how far into the future a traffic

rate process can be predicted for a given error constraint and what the minimum prediction error is over a specified prediction time interval. Another study (Papagiannaki, Taft, Zhang, & Diot, 2005) introduced a methodology to predict operations that have to take place in an IP backbone network. The presented traffic prediction model in the study on IP network is quite similar to the prediction in the web service system. However, capacity and performance monitoring methods are not equal due to the internal state robustness. In addition, the data management in IP network is only partially comparable to web service systems.

A framework (Vercauteren, Aggarwal, Wang, & Li, 2007) has been introduced to provide both long-term (in days) and short-term (in minutes) predictions. However, to ensure customers are satisfied in normal operations on a web service, the required prediction time frame is several weeks or even months. Another study (Hellerstein, Zhang, & Shahabuddin, 1998) describes a statistical approach to characterizing normal system operation for time varying workloads in a web server. The study does not recognise the actual performance of the server system but assumes that it is fixed over time and well-known. However, the performance in quite many installations is not known and is not fixed but varies in terms of time depending on the configuration manner and the internal states of the system.

The dynamic resource allocation is a suitable method in some cases according to the problems shown in this study. However, it requires resources organised as pools, wherefrom/whereto such resources can be allocated and returned. The dynamic configuration management can be an effective method to facilitate momentarily. Conversely, the total amount of resources cannot be increased or especially decreased flexibly by means of cost savings.

Performance engineering and testing oriented blog (Podelko, 2007) is professing that performance testing theory is almost non-existent and performance testing practice is pretty mature. The conception of the aforesaid blog is that activities are grouped around commercial tools and user communities. On the age of the mainframes, capacity planning was typical for corporations using mainframes. In another blog, Podelko (2009) pointed out that test vendors are on the easier side, "*trying to implement something to make it easier for an inexperienced person to create and run load test scripts*". While this produces nice analysis graphs and

may be helpful in some cases, it cannot replace performance engineering processes and good performance engineers.

1.3 Aim of the Research

The aim of the research is to enhance the previous practices using historical web service access information, combined with resource utilisation via natural load usage simulation. The intuition behind the approach is to use mathematical tools to process historical information and extract trends in the usage evolution at different time scales. The aim can be divided into three sub-objectives.

The first objective of this study is to introduce a regular basis prediction model for the sake of obtaining a web server system performance by using a known natural workload. This objective includes the following research question:

> **Question 1:** *Is it possible to measure the performance of the website system on a regular basis?*

The study introduces an entirely novel approach to predict the real performance of an e-commerce website system. The natural workload has to reflect the characteristics of past access. This model is also suitable for monitoring deterioration in performance.

The second objective of the study is showing that the actual usage of a web service is measurable and analysable automatically on a regular basis (for example, daily or weekly). This objective includes the following research question:

> **Question 2:** *Is it possible to analyse the usage of a website and characterise it automatically and on a regular basis to identify the peak load?*

The third objective of the study is showing that a gap between performance and peak usage levels can be seen as a spare performance of the system. This objective includes the following research question:

Question 3: *How can the current website performance and actual usage be compared to each other?*

The above research questions lead to the following research hypotheses:

Hypotheses

The total performance of the server system consists of a unique combination of hardware, applications, configuration, and status of different application layers. This study hypothesises that *the actual performance of the server system can be defined using a known natural load with a short test-time period*. During the test load, the response time and different resource utilisation indicators of the system are recorded into separate log files.

In this study, it is assumed that *the actual usage of a web server system is measurable and analysable automatically and the result can be exploited to construct the natural load test for simulation purposes*. The analysis can be based on ordinary access log files. The analysis can be done in a specified form in all cases, and the result is comparable despite the nature of a web service.

Furthermore, by *combining the analysed actual usage data and the calculated total performance of the web server system, the moment when the performance of the system runs out can thus be estimated*. This is the moment, when the system does not have enough performance to serve users within the required response times.

1.4 Research Approach

Design science creates and evaluates IT artefacts that solve the problems in the organization. The scientific aspect of design science answers the questions: can we build innovation and how useful will it be for the organization? We can also ask what kind of innovation should it be and how should it be created. The study by Järvinen (2004) suggests that if the research problem includes the following verbs: construct, alter, improve, create, repair, etc., it most probably belongs to design science. Hevner et al. (2004) believe that the result of design science is the artefact

itself. The artefact will provide a solution to the problem, and its research should provide new data on the topic in a new and innovative way.

IT-artefact is constructed to perform a specific function (March & Smith 1995), which indicates that completion of the construction of design problems has been resolved. Artefact construction action is aimed, in their opinion, at the advantage or value of the artefact produced by the user community. Van Aken (2004) suggests that design science is intended to provide information to a construction problem which can be solved, or the performance of existing systems that can be improved. Van Aken believes that innovation in the utility should be evaluated sooner rather than later.

Hevner believes that the results of design science are of four types: conceptual, models, methods, and realization. The concepts form a terminology of the research problem. Models indicate the relationship between concepts. Methods are steps while implementation of the artefact in the environment is realization.

This study, using design science in reference to Hevner, created seven guidelines for IT artefact design, implementation and evaluation. Instructions are summarized in Table 1, and their relevance is discussed in more detail in the text. The contribution of this study is proposing an analysis and forecasting model, which is suitable for optimizing website performance.

The first of these Hevner guidelines is IT artefact, which is intentionally built for one of the key problems of the organization. It can be implementation but also concepts, designs, or a method which has been applied to the construction and operation. IT artefacts are rarely completely ready for information systems, rather, they are the innovations that define the ideas, practices, technical capabilities and products that enable systems analysis and design; implementation and use can be effectively and efficiently implemented (Hevner et al., 2004). Hevner excludes people and the various elements of the organization, the IT artefact definition as well as the development during time. Implementation of an artefact is an indication of the design process and the outcome of the operation (Järvinen, 2004). This study of IT artefact is suitable for website performance analysis and forecasting model. An artefact of implementation is shown by designing and

constructing a website performance analysis and forecasting model, using a reference pattern in Hevner's seven IT artefacts for planning.

Table 1: Design science research guidelines, adapted from Hevner et al. (2004)

Guideline	**Description**
Guideline 1: Design as an artefact	Design science research must produce a viable artefact in the form of a construct, model, method, or an instantiation.
Guideline 2: Problem relevance	The objective of design science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design evaluation	The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research contributions	Effective design science research must provide clear and verifiable contributions in the areas of design artefact, design foundations, and/or design methodologies.
Guideline 5: Research rigour	Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.
Guideline 6: Design as a search process	The search for an effective artefact requires utilizing every available means to reach the desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of research	Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Design science emphasizes the importance of the research problem from a business perspective. Computer science research is about acquiring knowledge and understanding that enable technology-design and implementation artefacts so far unresolved or poorly resolved in the business of the problems. Hevner et al. (2004) points out in the second guideline that research is relevant, if it helps solve the problem of the community using it. As to a business perspective, the research is

important and relevant if the company stakeholders' requests for a website service to a high level, and on the other hand, it must be cost-effective.

IT artefact utility, quality and impact should be addressed through an accurate assessment and evaluation methods. The evaluation shall be based on the business environment demands artefacts, and it will be integrated into the IT infrastructure. The following features are used in the evaluation instrumentation: functionality, completeness, consistency, accuracy, performance, reliability, availability, suitability of the organization, and other necessary qualities. Iterative evaluation of the artefact, Hevner's third instruction, provides feedback on the construction of both process and outcome (Hevner et al., 2004). This study of IT artefact assessment focuses on how well the selected research methods are suitable for website performance optimization in different technological environments.

Hevner's fourth guideline affect the planning of design science research and will provide clear benefits in the following areas: the planned artefact, knowledge of the construction, the design of the assessment data, and methodology used. An important question in each case of the design science research is what kind of new and innovative landmark contribution that is delivering results. Design science research includes three different types of contribution subjects, at least one of which must be found for each study design. The first contribution is the subject of IT artefact itself. Artefact is the answer to the problem studied. It can be a solution to the problem studied, or it can generate substantial new scientific information on research or apply existing knowledge in a new and innovative way. The second area of contribution is the planning of construction process and modelling of the artefact. A significant contribution can be achieved, for example, in a new type of design process or model development. The third area of contribution is methodology. The method used in the study and the method for evaluating the research area itself will bring its own contribution. The evaluation methodology and metrics are, in the planning study, important aspects in themselves. The research contribution of this study is the extent to which the selected method is suitable for solving this research problem. In the future, it would certainly benefit those seeking a research method to study a similar problem (Hevner et al., 2004).

The study's scientific accuracy according to Hevner's fifth guideline must be proven by means of exact research methods as well as the IT artefact construction of the assessment (Hevner et al., 2004). Scientific research indicates the level of accuracy and how it is carried out. The design science research as well as accuracy is defined as the existing research knowledge and theoretical foundations of research methodology, not to mention its effective use.

In the sixth guideline, a good design approach is the search process using the available means to achieve the objectives of the study environment; however, by following the prevailing laws. Available measures and solutions Hevner et al. (2004) calls as a means to construct a solution.

Finally, the seventh instruction (Hevner et al., 2004) says that the results of the investigation should be forwarded to the management, as well as the technically minded people in the organization. The artefact, which has been described in sufficient detail presented technical personnel and practitioners, as well as explained how it is constructed.

In this study, the contribution of design science is achieved by examining how well Hevner's seven guidelines for creating an IT artefact suitable for website performance analysis and prediction model are set up.

1.5 Research Methods

At the beginning of the study, the actual log files are analysed using mathematical models. The study uses existing works on characterization of the load and pattern recognition in the different time frames measured by a number of requests and file sizes. In Figure 1.3, this phase is shown as actual usage analysis. The long-term trend is evaluated based on the log file analysis using time series models.

In addition, the real response times and resource utilisation are measured using natural load. The effect of adjustment rate on the load is studied, and while load settings are kept unchanged, results are comparable in the long-term. However, if the load is adjusted based on actual usage using log file characteristic, it allows for

a more precise view of the service performance. The real performance of the web system in a lengthy period is predicted by extrapolating from the workload information and systems load metrics, using the statistical pattern recognition. The natural load is marked as a load test in Figure 1.3.

During the natural load, critical server resources are monitored using SNMP (Simple Network Management Protocol) based on RFC 1157 (Case, Fedor, Schoffstall, & Davin, 1990). The monitoring of resources under the natural load facilitates the obtainment of raw data from the system internal status. The data can be used to predict the total performance of the system in the future. The operating system behind the SNMP interface is not discussed in this study. It is assumed that the SNMP response from the system produces valid and comparable results despite the differences in operating systems.

When a server system resource utilization is loaded for the upper limit or the maximum response time is exceeded, then the server system's overall performance limit is reached.

Figure 1.3 presents *Predictus*-model using BPMN notation (OMG (Object Management Group), 2011). On this basis, the method has been developed, which is implemented by several appropriate programming languages and simulation applications. Only the essential aspects of the method have been opened at a detailed level; unimportant parts are excluded from the consideration.

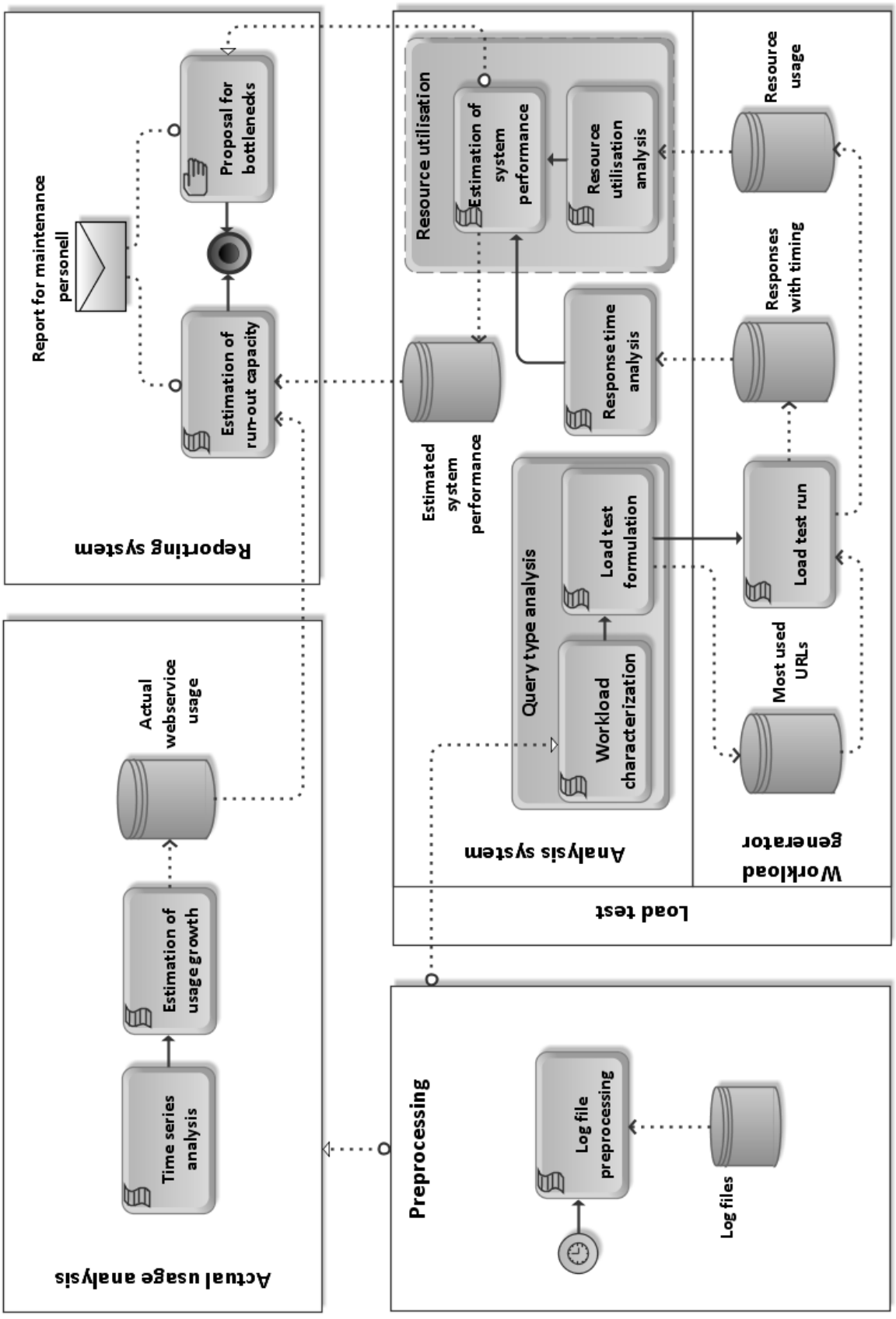

Figure 1.3: Performance estimating model, Predictus for distributed e-business architecture

The moment, when the performance does not reach the desired level, is possible to predict by combining the results from the analysis of the access log and the natural simulated load. In Figure 1.3, this is shown as an estimation of run-out capacity. The result of the evaluation is shown in a simple time-based graph that can be interpreted by non-performance experts. The bottlenecks can be recognised based on system load analysis.

1.6 Scope of the Research

In this study, the web server system is explored as distributed computing. Such computer systems, where the resources are divided among many virtual computers, are mostly seen as independent computer systems, which can be evaluated using the methods presented in this study. However, the virtualization specific questions are not discussed in this study.

Some non-predictable load peaks (hotspots), such as breaking news in media services, present an unexpected peak of usage, and are not discussed in this study. Instead, the gap between normal usage and maximum performance of the system should be continuously watched. If the potential user population is known, the resource consumption estimation in any unexpected situation can be calculated.

In this study, a simulation model is presented to define the actual performance of the system using simulated user requests. System load is monitored during the natural load to obtain the consumption of the system resources. By varying the system load, the change in response time and resource utilisation can be determined. The extrapolation method is used to expand the regression analysis results in order to predict the time when performance is not enough to satisfy the load required by real usage.

1.7 Related Studies

Some recent news (Aalto-www, 2010) from Aalto-university related that energy effective data centres are to be developed in Finland in a research project called *Energy-efficient server centres for Finland*. The aim of the project is to analyse ICT equipment and cooling systems in data centres and to figure out the best solutions to optimise energy consumption and utilise the waste heat.

FIT4Green (Fit4Green Consortium, 2010) aims at contributing to ICT energy-reducing efforts by creating an energy-aware layer of plug-ins for data centre automation frameworks so as to improve energy efficiency of existing IT solution deployment strategies. This is to minimize overall power consumption by moving computation and services around a federation of IT data centre sites.

1.8 Contributions of the Work

I worked as a data communications expert in a software company in the 1990s. At that time, client-server systems, which were based on proprietary protocols, was generally used for commercial purposes. Computer equipment was expensive then, and cloud services were not even discussed. I have encountered regular problems in the performance of a variety of systems, particularly the fact that they had to be resolved ad-hoc. It turned out that the current problem could be solved with hard work, but a long-term solution did not exist. I then came up with the idea over whether it would be possible to track the performance of information systems as with any other company's operational system or activity. And in particular, I became interested in it, especially the manner in which performance can be managed proactively. At that time, an idea was born, namely, that the concept of performance can be solved by means of an impulse-response pair.

No ready-made solution or even one in the right direction, the idea could not be realised in only a number of years. But the idea of proactive performance management model began to mature gradually over the years. Over time, proprietary protocols have decreased, while the http protocol gained ground. It

provided much better opportunities for the development of performance testing tools. When I was in the central role in a web development and maintenance work in the early 2000s, there was a need to develop a model that works in practice. At that time, I noted that the performance management tools were developed for web services, but they were merely reactive "fire fighting," or were intended to improve the performance of application code. They did not make it possible to monitor the actual use of the service and in particular, to anticipate the future use or application behavior. In other words, the impulse (the actual website usage) and response (the resulting system load) were completely unknown to the performance monitoring process. They also did not support the continued development of website applications and continuous testing.

Right from the beginning of model development, it was clear that the performance analysis and monitoring should be a continuous operation, in which case it must be as highly automated as possible. From this perspective, I began to develop a method to ensure website performance throughout its life cycle. I developed a method comprising the following necessary key components: the identification of system components, test load management, the related data collection, the analysis of results and their interpretation and follow-up.

Initially, the process was developed in two separate websites. It could be generalized to very small changes and required only one way to describe the various layers of hardware and the number of servers and their key components. After that, it was found that the model can be generalized to almost all commonly used HTTP and HTTPS websites.

In this work, the hardware identification details have been overlooked and are only focused upon in order to consider the core of the model alone. The generalized *Predictus*-model has been able to ascertain that the application components are related to the method and work well in different environments, and that the analysis produces reasonable and verifiable results in different test sessions.

The functionality of the *Predictus*-model has been verified in practice in very different environments. Differentiating factors have been the applications, the website's purpose and scale of use, a different hardware architecture distributed

among a wide network of hardware or a single service on a dedicated hardware. Differentiation was tested using the characteristics of the tested item. Not only is the generation of load testing and monitoring independent but so are the results of the analysis platform.

A large number of different websites have been tested using the *Predictus*-model. Therefore, this study is based on a large number of log files, and a very large number of test runs in a number of e-business environments. In this study, the data is mainly derived from four different independent websites. Of these websites, architecture is discussed later in Chapter 5 .

In this study, the performance measurement and analysis of the model are a completely new kind of approach. It is based on my idea to combine well-known load test and control the load and burden due to the hardware components. The well-known test load, of course, corresponds as closely as possible to the load of real users. And since users cause the load changes with time, it is obviously to be taken into account for the long-term analysis.

1.9 Thesis Outline

The state of the art in modern e-commerce website technologies and in performance engineering is described in Chapters 2 and 3. Chapter 2 is focused on web service requirements from the viewpoint of users, the boosting technologies and some essential problems. Chapter 3 is focused on principles of performance management, characterisation of access log, focal concepts and principles, and finally the tools for monitoring and estimation.

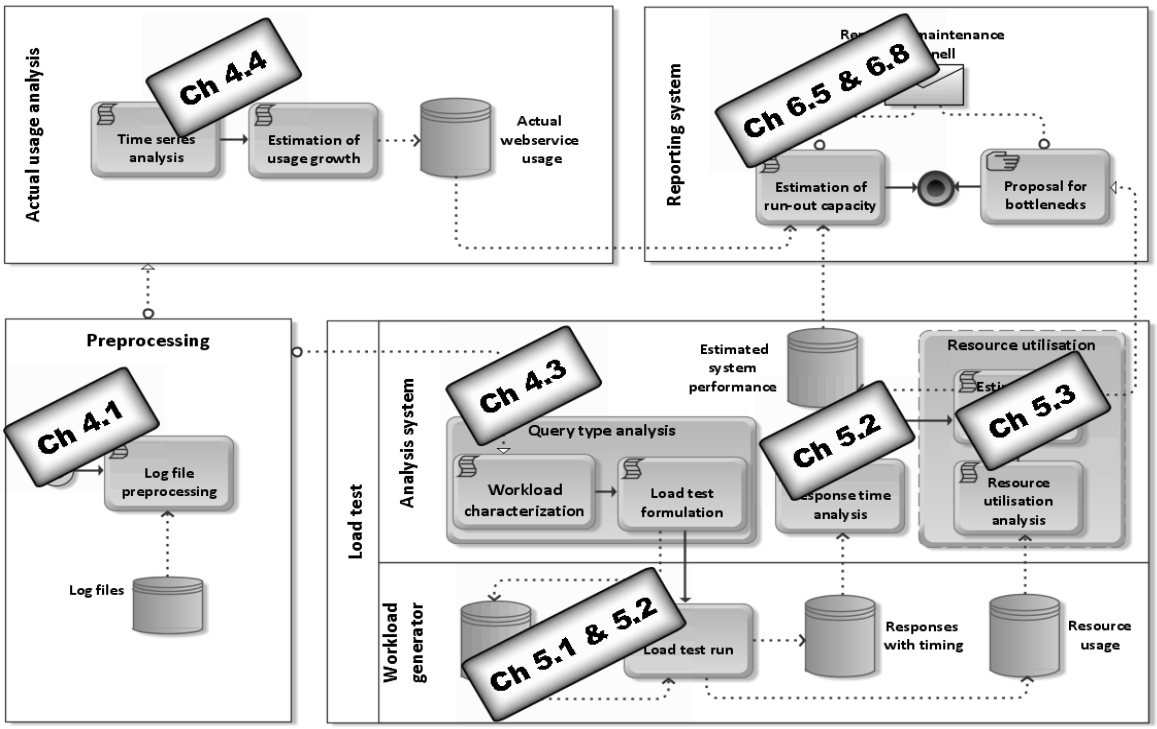

Figure 1.4: Structure of the thesis

The methods used in this thesis are discussed in Chapters 4 and 5. The analysis and characterization of a commercial website are described in Chapter 4. It answers the research question, how to actual usage of the website can be measured and analysed on a regular basis as well as the way in which the peak load forecast is formed during normal usage. The natural load, monitoring of the server resource consumption, and analysis are described in Chapter 5. It answers the research question, how the network hardware performance can be measured on a regular basis, and it will form a forecast of future performance. Figure 1.4 shows the structure of the method and the corresponding book chapters, wherein each component of the method described below.

Synthesis of the results of the actual usage analysis and natural load is put forward in Chapter 6. The actual performance of the web system and the analysis of additional resources are described in this chapter. It answers the research question of how the gap between current performance and the spare performance can be visualized easily understandable form.

Finally, a general discussion and conclusions round off this thesis in Chapter 7.

2 Website Environment

This chapter describes the requirements for a complete web service system, which generally consists of hardware, software, configuration, and applications. Some of those requirements are derived in order to satisfy user requirements while some are meant to satisfy system management requirements.

2.1 User Expectations

A website must be easy to follow, be consistent and predictable, and must seem simple and natural. If a website has actually delivered the information or service(s) the customer was looking for, a customer has found a reason to stay on the site. Users have several common requirements: performance, usability, navigation, and many others. Some of these requirements may be explicitly stated through formal documents, while others are implied or assumed. Some general statements are the following (Loosley, 2005):

- An *unreachable website* is useless for the user, despite good reasons to visit such a site.
- Having reached the site, pages that *download slowly* are likely to drive customers to try another site.
- If the site is sufficiently responsive, *other design qualities* come into play.

Web users' tolerance for loading delays depends on several factors, including expectations, site feedback, the complexity of a task, importance of the aim, and the relevance (utility) of the information being provided by the site. And their perception of a site's quality and credibility diminishes as its download times increases.

Selvidge et al. (2002) studied the variable impact of web delays on losing users, frustration and proportion of task completion. The current HTTP1.1 technology does not allow users to get an estimate of the amount of time they would have to wait when downloading a document but only provides a real-time measure of the

amount of content that has been downloaded. Sometimes, users wait for a significant amount of time before being refused connection. This is quite frustrating for users and may lead to avoiding a specific site. Hence, for website designers it is important to design the website in a manner that neither the waiting time for a requested connection nor the chance that the user is refused connection will be too high.

If users are just browsing through the internet for entertainment, they may be more tolerant of download delays than if they have to find the information to complete a task before a certain deadline (Selvidge et al., 2002). Another important aspect that influences tolerance for time delays is whether the information was worth the wait or if the information was considered valueless. Tolerance for delays could also be related to tasking demands, web page content, or attributes of the population of users sampled, such as computer-experience level and download speed they typically experience. Some old studies (Kuhmann, 1989; Kuhmann, Boucsein, Schaefer, & Alexander, 1987; G. Martin & Corl, 1986; Weiss, Boggs, Lehto, Shodja, & Martin, 1982) have shown that longer delays or system response times increase frustration and stress, and decrease productivity. Galletta et al. (2003) have shown that decrease in performance, attitudes and behavioural intentions is not necessarily linear.

The subjective understanding of adequate or good performance has to be defined using measurable and communicational means. The problem exists especially if the system for which we are considering this question does not exist as yet. The problem is that of setting performance requirements for an as yet non-existing system. Typically, the requirements are specified in a non-quantitative way. Statements such as the following may be part of requirements: The system should have low overhead, the memory and processor speeds should be synchronized, there should be a low probability of failure, and so on. In all of these cases, the qualitative requirements are stated, which may be quite difficult to measure and realize. They are non-specific, non-measurable, and therefore, unacceptable. To change this, the analyst should look at what the system will be required to do, and what performance would be needed for a typical system with the same loads.

The basic advice regarding response times has remained unchanged for sometime now (Miller 1968; Card et al. 1991; Menascé & Almeida 2002) :

- 0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result.
- 1.0 second is about the limit for the user's flow of thought to stay uninterrupted, even if the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 and less than 1.0 second, but the user does lose the feeling of operating directly on the data.
- If the response time for the transaction exceeds four seconds, but remains below six seconds, then 60% of the search transaction will be lost because users will abort the search, and potential sales will be lost.
- If the response time exceeds six seconds at the web server, then 95% of the search request will be aborted.
- Normally, response times should be as fast as possible, but it is also possible for the computer to react so fast that the user cannot keep up with the feedback.

A study (Galletta, 2002) has shown that relatively small increases in delay can have a profound impact on how users react to websites. Lindgaard & Dudek (2002) found that user satisfaction is a complex construct comprising several affective components as well as a concern for usability. User satisfaction in the context of business to consumer (B2C) websites is a complex construct comprising concepts that cannot all be captured under the term ‘satisfaction’. Concern for usability as well as expectations based on interactive experience are integral to the experience, although usability appears to be assessed independently of affective aspects of user satisfaction, at least in browsing behaviour when the interaction is not hindered by severe usability problems.

A user expects technically constant quality from the service even if there are a number of simultaneous users. The technical service level quality indicator is simply measured by the user-experienced response time. An earlier study found

that waiting time is the most objectionable deficiency of the medium (Lightner & Zeng, 2009). The user-experienced response time is determined by the weakest link in the chain from the server to the browser: the throughput of the server, the server connection to the internet, the internet itself, the user's connection, and the rendering speed of the user's browser. Only the server throughput and the server connection to the internet are maintained by the web service provider. Sufficient system resources are needed to keep the quality of service predictable.

2.2 Fault Tolerance

To ensure the availability of a web service, it should be protected against a single point of errors. Tanenbaum and Van Steen (Van Steen, 2003) explained that a system is fault-tolerant if it maintains four characteristics: *availability*, *reliability*, *safety*, and *maintainability*. Availability ensures that the system is available to be used at any given moment. Reliability allows the system to perform continuously without failure. Safety makes sure that in the event of failure, it should not cause any undesirable behaviour that compromises the safety of the system and its users. Maintainability measures how straightforward a failure can be repaired.

Tanenbaum and Van Steen (Van Steen, 2003) classify the different types of failings that can occur:

- Communication failure happens when a message may be delayed, lost, or corrupted. The server may be crashed after receiving a request, and the client may be crashed after sending a request.
- The Byzantine failure occurs when a server does not behave in the correct manner. For instance, a server produces an output which it should not produce or does not produce any output at all.
- Omission failure occurs when a server fails to respond to a request based on different reasons such as connection failures. Crash failure is a subclass of omission failure. It occurs when a server systematically omits all outputs and nothing is heard from that server anymore.

- Timing failure occurs when a server either omits the specified output or responds too early or too late. In the situation where the server responds too late, it will affect the performance of the system.
- Response failure happens when the server's response is incorrect. The server might provide the wrong reply to a request, or it reacts unexpectedly to a request.

There are a number of techniques that can help achieve fault tolerance in a web service. The most common technique is redundancy. Physical redundancy deals with hardware or software; it runs an extra hardware or software at the same time to provide the correct output. In the case of a web service system, it leads to distributed computing. Distributed computing is a common way in web server systems to guarantee the fault tolerant, but it also leads to better scalability. However, increasing the number of servers is also increasing the maintenance costs such as energy consumption, maintain costs, licensing fees, etc. Hence, the amount of servers has to be optimised between scalability, fault tolerance and the economy.

2.3 Cluster Computing

Distributed system is defined in an old study (Booth, 1976) as a collection of computers, which are remotely located from a central computing. On some other studies (Mullender, 1993; Van Steen, 2003), the distributed system is seen as a loosely coupled, autonomous computer system with their own failure modes, which can execute logically separate computations.

In general, cluster computing is the technique of linking two or more computers into a network in order to take advantage of the parallel processing power of those computers. There are several varieties of computer clusters, each offering different advantages to the user. These varieties are considered in the following.

High-availability (HA) clusters

High-availability clusters are also known as fail-over clusters. They are designed to ensure non-stop access to service applications. The clusters are designed to maintain redundant nodes that can act as backup systems in the occurrence of a defect. The minimum number of nodes in a high-availability cluster is two – one active and one redundant – though most high-availability clusters will use considerably more nodes. High-availability clusters aim to solve the problems that arise from mainframe failure in an enterprise. Rather than losing all access to IT systems, high-availability clusters ensure non-stop access to a system. This feature is especially prominent in business, where data processing is usually time-sensitive.

Load-balancing clusters

Load-balancing clusters operate by distributing a workload evenly over multiple back end nodes. Typically, the cluster will be configured with multiple redundant load-balancing front ends. Since each element in a load-balancing cluster has to offer full service, it can be thought of as a high-availability cluster, where all the available servers process requests.

Load-balancing clusters operate by routing all requests through one or more load-balancing front-end nodes, which then distribute the workload efficiently among the available active nodes. Load-balancing clusters are highly useful for those working with limited IT budgets. Assigning a few nodes to managing the workload of a cluster ensures that limited processing power can be optimised.

High Performance Clusters

High performance clusters (HPCs) utilize the parallel processing power of multiple nodes. They are commonly used to perform operations that prefer nodes to communicate as they perform their tasks. The best known HPC is Berkeley's Seti@Home Project, an HPC consisting of over 5 million volunteer home

computers applying processing power to the analysis of data from the Arecibo Observatory radio telescope.

Generally high performance clusters are used primarily for computational purposes, rather than handling IO-oriented operations such as a web service or databases. For instance, a cluster might support computational simulations of weather forecasts or vehicle crashes. The primary distinction within computing clusters is how tightly-coupled the individual nodes are. To illustrate, a single computed job may require frequent communication among nodes - this implies that the cluster shares a dedicated network, is compactly located, and probably has homogeneous nodes. The other extreme is where a computing job uses one or few nodes, and needs little or no inter-node communication. This latter category is sometimes called "Grid" computing. Tightly-coupled computed clusters are designed for work that might traditionally have been called "supercomputing".

Grid computing

Grid computing is optimised for workloads, which consist of many autonomous tasks or packets of work that do not have to share data between the jobs during the computational process. Computer grids serve to manage the allocation of jobs to computers, which will perform the work independently of the rest of the grid cluster. Resources such as disk storage may be shared by all the nodes, but intermediate results from one job do not influence other jobs in progress on other nodes of the grid.

Characteristics of a distributed system are described in other study (Van Steen, 2003) as follows:

- By *transparency*, to the user of the system it appears that all tasks are handled by a single computer.
- *Scalability* ensures that the system will still be able to perform all user requests without any performance degradation by adjusting the number of servers.

- *Fault tolerance* enables the system to continue to operate regardless of failure, and that users are unable to notice these faults.
- *Openness* makes it possible to improve and extend from the original system without the need to restructure the entire system.
- *Concurrency* allows servers to handle different requests from multiple clients simultaneously.
- *Heterogeneity* guarantees, that each server on a distributed system may have different hardware and different software versions.
- *Security* provides a secure communication channel for the users and ensures access to authenticated users only.
- *Resource sharing* provides users the ability to access resources anywhere in the system.

Generally, distributed systems aim to perform without being seriously affected by any failings that might occur in the system. If there is failure, the system is still able to recover and users would be unable to know this failure.

An effective and a most common way is to combine multiple Web servers called "clustered Web server" or "server farm" and balance the load among these servers. In order to address the network latency delays caused over greater distances, large organizations are also deploying distributed Web servers in different locations. User service requests are routed to a server based on some routing algorithms. The system performance depends critically on these routing algorithms. This method of load management has been shown to improve the quality of service (QoS) in practice and is thus widely used. One advantage of using multiple servers is that one need not develop very accurate plans for the server capacity; one can add to the existing capacity in an ad-hoc fashion through either new servers or by employing unused capacity elsewhere (Zhang & Fan 2008).

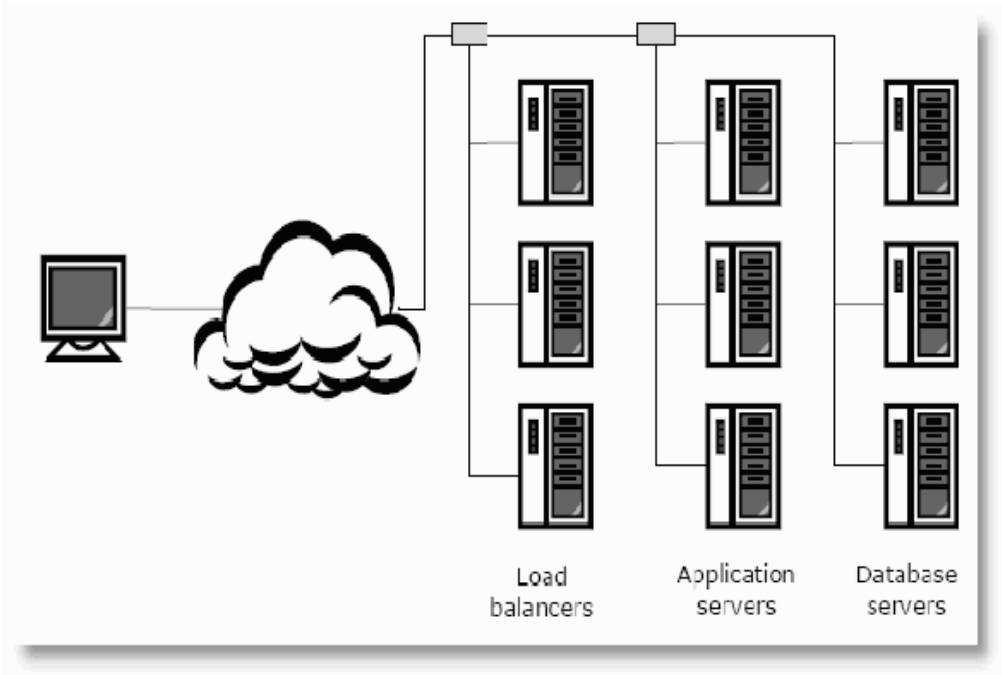

Figure 2.1: Architecture of clustered system. The server farm consists of several dedicated server clusters

The clustered system architecture, where requests are served by the system through different clusters of servers, from the web server cluster to the application server cluster, and possibly to the database server cluster as shown in Figure 2.1. In general, it is possible for requests to be served by a subset of these clusters. Even though this infrastructure is architecturally simple, the system is quite complicated with a load balancing mechanism. It consists of numerous clusters of servers, each of which can have quite a number of software and hardware components. A typical Web system is comprised of several nodes with tens of applications running on them. Given the great complexity of the overall system, planners are constantly confronted with questions regarding: how many servers to place at each cluster in the current infrastructure; what layout can deliver the best QoS; is there enough performance available to support the expected business possibilities and future growth?

2.4 Web Service Performance Boosting Technologies

The Web traffic has experienced huge growth in the last decade. Content providers and e-commerce merchants are overwhelmed in many cases by the number of request for web pages resulting in considerable deterioration (for example, long response time) in web server performance. In order to keep the response time within a satisfactory level, administrators limit the number of simultaneously open connections. When the number of request overtakes this limit, all later user access requests will be rejected. Hence, often either the waiting time is high or the user request is not processed at all. Either situation could lead to disappointed customers, forcing many of them away from the site. The traffic may change with the time of the day, the day of the week, or even the month of the year. These seasonal or periodic fluctuations make it more difficult to come up with a proposal over the adequate performance of a site.

There are various solutions developed to solve this concern, such as replacing servers with more effective ones, caching or outsourcing. However, these approaches to load management have their own troubles. For example, a server replacement may work for a while, but it is not scalable and could cause interruption due to server upgrade and maintenance. Besides, if the server performance is planned based on the peak load, then the added performance is useless during the off-peak hours. Outsourcing, on the other hand, has a certain price tag, and yet one has limited control over the Quality of Service. Caches have been used in several ways to address the issue of slow response time caused by overloaded servers. The caches intercept requests for Web content, and attempt to respond to the requests whenever possible. When these requests cannot be served from the caches, they are forwarded to the Web server. The presence of dynamic content featured on most websites raises significant barriers to caching.

Many websites dynamically generate responses on the fly when user requests are received. One study (Titchkosky, Arlitt, & Williamson, 2003) has empirically evaluated the impact of three different dynamic content technologies (Perl, PHP, and Java) on web server performance. The results of this study show that the

overheads of dynamic content generation reduce the peak request rate by up to a factor of 8, depending on the workload characteristics and the technologies used.

2.5 Software Aging and Rejuvenation

Some studies have reported the software aging phenomenon (Garg, van Moorsel, Vaidyanathan, & Trivedi, 1998; Huang, 1995) in which the state of system performance degrades slowly. The primary symptoms of this degradation include exhaustion of system resources, such as memory leaking, unreleased file locks, data corruption, and instantaneous error accumulation. This may eventually lead to performance degradation or other unexpected effects. A proactive fault management method to deal with the software aging phenomenon is software rejuvenation. Unplanned computer system outages are more likely to be the result of software failures than of hardware failures (Gray & Siewiorek, 1991; Sullivan & Chillarege, 1991). In some studies (Grottke et al. 2006; Silva 2006), software aging has been reported widely encountered as well as in high-availability and safety-critical systems. This essentially involves gracefully terminating an application or a system and restarting it in a clean internal state. This process removes the accumulated errors and frees up operating system resources. The preventive action can be done at optimal times (for example, when the load on the system is low) so that the overhead due to planned system downtime is minimal. This method, therefore, avoids unplanned and potentially expensive system outages due to software aging.

Aging is explained in Castelli et al. (2001) as caused by software that is extremely complex and never wholly free of errors. In a web application environment, it is practically impossible to completely test and verify that a piece of software is bug-free. This situation is further exacerbated by the fact that web software development tends to be extremely timed to market driven factors, which results in applications that could meet the short-term market needs, yet do not account very well for long-term ramifications such as reliability. Hence, residual faults have to be tolerated in the operational phase. These faults can take different types, but the ones that we are concerned with are the causes of long-term exhaustion of system

resources such as memory, threads, and kernel tables. The essentially economic problem of developing and producing bug-free codes is not the problem at hand; rather, we are addressing one of the problems that arises from the prevailing approach to developing software, and one approach to attacking that problem is software rejuvenation.

The following Figure 2.2 and Figure 2.3 from Avritzer et al. (2002) provide an example of software aging. Figure 2.2 shows throughput versus time of a web server application running on a Unix server. The test started at time 0 but about 5½ hours into the test, things went awry. Throughput, which had been holding steady at 62 transactions per second, suddenly fell effectively to zero. Figure 2.3 shows the memory profile for the Web server process during the test. Initially 13 MB, the process size grows rapidly at first, and then steadily until reaching 182 MB at the 5½ hour point of the test. At this point, the process stopped growing, but also stopped serving all but a very small number of transactions. The other transactions' timed-out were discarded by the load generator. This normal growth, which reaches a certain level and stops, needs to be distinguished from a fatal memory leak, in which memory grows to the point that something breaks and the application fails to function.

Memory leaks are a common software fault (Avritzer et al., 2002) in applications written in languages in which the programmer is responsible for memory allocation and recovery. Memory gets allocated but never gets freed due to a fault in the code. A memory leak is considered to be fatal if it results in the application crashing or else failing to function. Aging anomalies can be detected in load test by executing a long soak run with a fixed workload while monitoring application throughput and resource consumption.

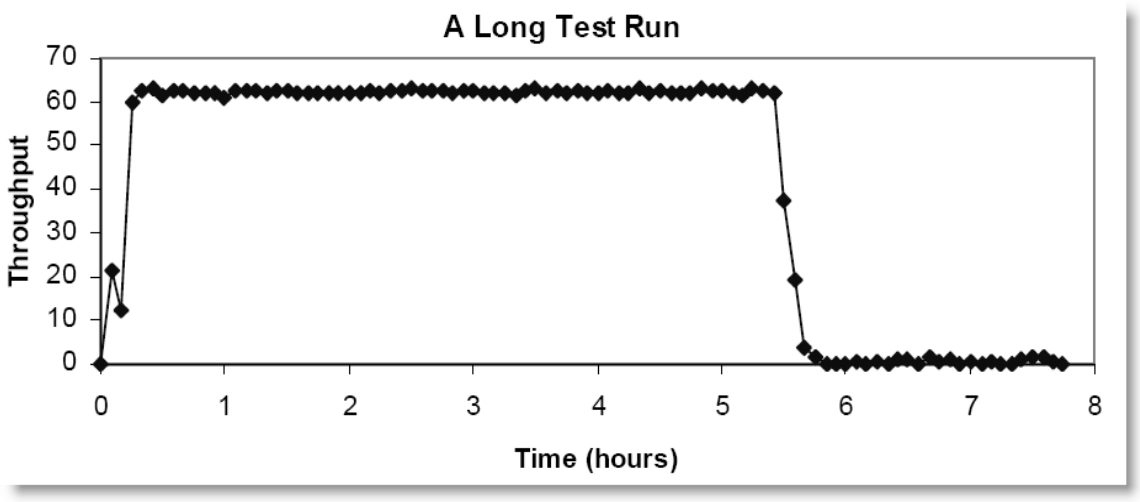

Figure 2.2: Throughput deteriorating in system with fatal memory leak. The throughput is maintained in a stable level until the available memory reaches zero at 5½ hours.

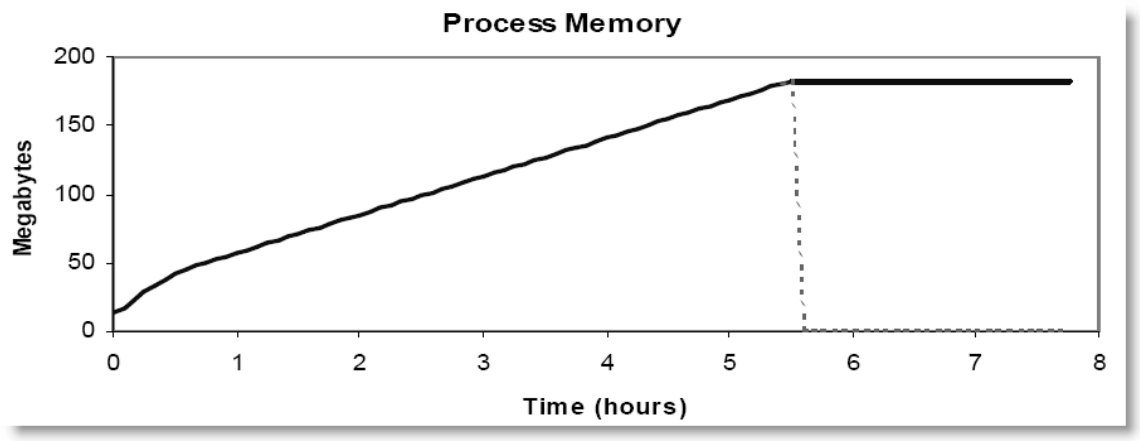

Figure 2.3: Memory usage in case of fatal memory leak. The available memory is decreased until it reaches zero at 5½ hours.

Menascé et al. (2003) explained that the speed of aging is dependent on the load on the server. With a higher load rate, the aging process is advanced faster. The load dependent aging process is formulated as follows:

$$\lambda_{ag} = C \times (\bar{n}_s)^a \times \lambda_{ag}^i \quad (1)$$

Where *C* and *a (a>0)* are constants, $\bar{n}_s$ is the average number of requests at the server, and λ_{ag}^i is the load independent aging rate. In the case of C=1 and a=0, the load independent aging rate is expressed.

The system performance degradation also occurs due to some other factors. Thalheim & Tropmann (2011) suggest that even if the system is working effectively in the initial stages, the performance might be reduced if the system load does not change. This may be due to the increase in the size of the database tables or even due to other competing systems.

2.6 Conclusion

A website has to be reachable as expected by users. Availability is aimed to keep as high a level as possible in e-commerce sites. The availability rate is aimed to be increased using redundancy or other fault tolerant techniques. However, complex redundancy produces more complexity in the analysis.

Cluster computing is a highly effective way to achieve a fault tolerant website environment. However, the fault tolerance creates some challenges for analytical methods. At first, while one server from a server farm is in a malfunction state due to hardware or software failures, the analytical models have to be updated immediately to become a valid analytical model. Secondly, all the redundant servers have to be similar, otherwise the models would be laborious to maintain.

The performance models are complicated due to the web performance boosting techniques. For example, the usage of cache produces uncertainties, when the response is achievable from memory and when the much slower I/O-oriented response is required. The difference in response time is remarkably higher in case of disk search instead of memory search. In an analysis, it cannot be assumed to benefit either.

Software aging is responsible for the continuous fluctuations on the performance of the server farm. Slow software aging is hard to perceive and the possible damage is immediate in case of resource run-out. In the worst case, the throughput can bring down the complete web server system without prior warning and then rejuvenating can take several hours.

3 State of the Art in Performance Analysis

This chapter provides a review of web access log analysis and performance management. The traditional performance management is focused on defining the throughput of the present server system. The access log analysis is the past usage of a service, e.g. how many requests at a time, and what is the size of response, etc. Normally, the analysis is done by examining the log files of the web application.

3.1 Access Log Analysis

Typically, workload is exposed with linear extrapolations or other curve fitting methods. While such techniques addressing simple trends, they do not capture different time frame variations. One research (Hellerstein et al., 1998) describes an approach to characterization of the web server access log. The methodology of the research can capture periodic effect (e.g. time-of-the-day and day-of-the-week) and trends (e.g. growth in user demand from month to month). Having a statistical model that characterizes normal behaviour allows for extrapolating values of metrics (e.g. HTTP operations) so that three questions can be answered: (1) What will the workload be at a specific time in the future; (2) When will the workload grow beyond a specific limit; (3) When will this limit be exceeded during a specific time-of-the-day or day-of-the-week. The limit of the research was seen as a given value.

One of the methods is based on decomposing the usage data to daily, weekly, and monthly fractions using analysis of variance (ANOVA) technique, which is expressed as:

$$y_{ijkl} = \mu + \alpha_i + \beta_j + \gamma_k + \epsilon_{ijkl} \quad (2)$$

, where μ is a grand mean, α_i is a deviation of daily pattern, β_j is a deviation of weekly pattern, γ_k is a deviation of monthly pattern and finally ϵ_{ijkl} is an error term. The regular peak loads are exposed by data decomposition, and the

prediction relies on those peaks. The prediction is made using least-square regression.

The study by Papagiannaki et al. (2005) presented a methodology for predicting when and where the link maintain operations have to take place in the core of an IP network. The methodology is claimed as simple to implement, and can be fully automated. In addition, it provides accurate forecasts for at least 12 months into the future. Hence, it is suitable within the context of capacity planning. However, multi-resolution analysis (MRA) of the original signal and modelling of selected approximation and detail signals using ARIMA models could possibly provide accurate forecasts for the behaviour of the traffic at other time scales, such as from one day to the next or at a particular hour on a given day in the future. These forecasts could be useful for other network engineering tasks, like scheduling of maintenance windows or large database network backups.

Cleveland et al. (1990) expressed the method STL (Seasonal Decomposition of Time Series by Loess) for the decomposition of time series in terms of three components: trend, seasonal, and residual. Some details in design goal make the STL suitable for access log data decomposition. Namely: a) flexibility in variation in the trend and seasonal components; b) the ability to decompose series with missing value; and c) robust trend and seasonal components that are not distorted by transient, aberrant behaviour in the data. STL consists of a sequence of locally weighted smoothing operations. The implementation is based on computer routines. One of the implementations (Team, 2011) has been made effectively in R.

A study by Baryshnikov et al. (2005) investigates the potential for predicting hotspots sufficiently far in advance, so that preventive action can be taken before a hotspot takes place. Performing accurate load predictions appears to be a daunting challenge at first glance, but this study shows that when applied to web server page-request traffic, even elementary prediction methods can have surprising forecasting power. The study shows that there is useful predictability in internet traffic that can be applied to the use of resources that experience strong surges in traffic. Prediction algorithms in practice may be supplemented with partial prediction data such as the general timing of particular events. For example, it may

be known that an announcement will appear within a couple of hours on a given day, but the timing is otherwise unknown.

In another study (Lu, Yang, & Zhao, 2004), a prediction was made using MRA wavelets. Using wavelet MRA has been able to show the overall long term trend, as well as analysing variability at multiple time scales. The largest amount of variability in the signal comes from its fluctuations at the 12-h time scale. The analysis indicates that a parsimonious model consisting of those two identified components is capable of capturing 98% of the total energy in the original signal, while explaining 90% of its variance. The resulting model is capable of revealing the behaviour of the network traffic through time, filtering short-lived events that may cause traffic perturbations beyond the overall trend.

3.2 Forecasting of Workload

Coffman & Odlyzko (2001) said that everything changes so rapidly on the Internet that it is impossible to forecast far into the future. The internet has been increased at about 100% a year for its entire history. This observation could be used to extrapolate the growth rate into the future, and predict that traffic will continue to double every year. However, this assumption would hardly work for all websites. Due to the nature of the propagation on the Web, it is hard to predict the points in time at which hotspots will occur. This situation occurs in many types of websites. For example, an unpredictable stock market crash can generate a huge increase in traffic to sites with financial news and analysis. For practical purposes, it is valuable to predict the magnitude of possible hotspots even when timing cannot be predicted. Menascé & Almeida (2002) described two forecasting strategies, quantitative and qualitative. The former relies on the existence of historical data to evaluate future values. The latter is a subjective process, based on judgements, intuition, expert opinions, historical analogy, commercial knowledge, and any other relevant information. Qualitative analysis plays an important role in cases where little or no historical data is available. Quantitative analysis may rely on historical access log or any other regular business statistics, which can affect hotspots in the web services.

Menascé & Almeida (2002) collected several essential characteristics of forecasting. At first, it suggested that a good forecast is more than just a single number; it is a set of scenarios and assumptions. Time plays a key role in the forecasting process. The longer the time horizon, the less accurate is the forecast. Secondly, forecasting horizons can be grouped into the following classes: A) short-term (<3 months); B) intermediate term (e.g. from three months to one year); and C) long-term (>1 year). Demand forecasting in the Web can be illustrated by typical questions that come up very often during the course of capacity planning projects. Thirdly, there are good questions: Can we forecast the number of visitors to the company's website in order to plan the adequate capacity to support the load? What is the expected load for the credit card authorization service during the Christmas season? How will the number of messages processed by the e-mail servers vary over the next year? Finally, there is encouragement for the planning process. Implementation of Web services should rely on a careful planning process, i.e. a planning process that pays attention to performance and capacity right from the beginning. Planning the capacity of Web services requires a series of steps to be followed in a systematic way. One of its key steps is workload forecasting, which predicts how the system workload will vary over time.

From a load modelling point of view, the difference in using computing resources has changed the type of model for workload characterization. While in the early days of computing (70's) the typical systems were used in batch or interactive mode (Ferrari 1972), static workload models could adequately represent the user behaviour. In the 80's, dynamic workload models were introduced (Calzarossa et al. 1986; Ferrari 1983; Haring 1983) which were able to represent variabilities in user behaviour. In the 90's, generative workload models (Raghavan et al. 1993; Barford & Crovella 1998) have been proposed as a suitable method for capturing the dynamics and changes in the system.

Workload may change in various ways. Menascé & Almeida (2002) defined three different dimensions in e-mail traffic: the number of users, the number of messages per user, and the size of messages. Compounding the problem is the fact that the three dimensions expand at different growth rates. In one case, the number of messages observed had a threefold increase, but the size of messages

went up twenty times in the same period of time because of attached graphics. However, web workloads and traditional workloads change in different ways, depending on prospective business and technology evolution. The workload change happens for several reasons: new applications, increase in the volume of transactions and requests processed by the applications, enhancements of the application environment, marketing and sales promotions, and overall economic factors.

Most companies consider access logs to be very sensitive data. Hence, there are only few published studies (Arlitt et al. 2001; Menascé 2000; Menascé et al. 1999) of e-business workloads due to the difficulty of obtaining actual logs from website maintainers. In Menascé et al. (1999), the authors propose a graph-based methodology for characterizing e-business workloads and apply it to an actual workload to obtain metrics related to the interaction of customers with a site. For example, the paper shows how to obtain information such as the number of sessions, average session length, and buy-to-visit ratio. Menascé (2000) presented several models for workload characterization of e-business sites. It also shows how workload models can be obtained from HTTP logs. Arlitt et al. (2001) characterized the workload of an actual e-commerce site for the purpose of analysing its scalability. They use performance-related criteria to cluster requests into similar groups. They then use multiclass queuing models to carry out a performance planning study for the site. In Menascé & Almeida (2002), the authors study the effect of time scale on operational analysis for a large web-based shopping system. They show that time-related service level agreements and input parameters for predictive queuing models are sensitive to time scale.

Andreolini et al. (2002) suggested that most web benchmark tools work fine when used to analyse a single server system, but none of them is able to address all issues related to the analysis of distributed web server systems. The authors refer to tools like SURGE (Barford & Crovella 1998) and Webstone (Mindcraft, 2010). Such tools suffer also age problems, as they do not support dynamic requests and the recent protocols. The study summarised a lack in ability to sustain realistic Web traffic under critical load conditions, the difficulty or impossibility of emulating realistic dynamic and secure Web services, and the poor support in analysing

collected advanced statistic properties. As a consequence, the authors conclude that there is a lot of room for further research and implementation in this area.

The hierarchical and multi-scale characterization approach has been used in Menascé et al. (2003) to identify several characteristics in the workload of the two sites analysed. Some of the findings are:

- 88 % of the sessions have less than 10 requests.
- The session length, measured in the number of requests to execute e-business functions, is heavy-tailed.
- More than 70 % of the functions performed are product selection functions in contrast to product ordering functions.
- Requests to execute frequent e-business functions exhibit a similar pattern of behaviour as observed for the total number of HTTP requests.

Mahanti et al. (2009) has analysed a non-commercial, WWW2007 conference website. The datasets were collected over a 1-year period in 2007 in the form of access logs (server-side) and Google Analytics (Google Inc., 2010) (client-side) reports. The datasets contain approximately 10 million requests from about 130,000 unique visitors generating 215 GB of traffic volume. The measurements are used to characterize the usage behaviour of the website visitors. Modern websites (including WWW2007) contain a lot of graphics, including photos, banners, logos, maps, and menus. Hence, web pages and images account for approximately 75% of the total data. The rest of the traffic volume is attributable to visitors downloading PDF documents from the website. The average visit duration varied between 2 … 4 minutes, except for the last three months leading up to the conference when the average visit duration varied between 4 minutes and 6 minutes. Approximately, 70% of the visits lasted less than 1 minute. Most of the visits with page depth greater than 3 can be attributed to search engine spiders and conference organizers. Furthermore, single-page visits were mostly restricted to the homepage, the program page, the call for paper page, the important dates page, or specific paper downloads. The top 18 pages accounted for about 53% of the total page views. For obvious reasons, the most viewed page was the homepage at 19%.

The total traffic volume transferred was based on a day of week and hour of the day. The study found that the work week accounted for almost 80% of the total traffic volume. Each weekday has about 15 … 17% of the total traffic volume, with Monday being slightly busier.

Zhang & Fan (2008) has shown that content providers and e-commerce merchants are often overwhelmed by the number of request for Web pages and online transactions, resulting in considerable degradation (for example, long waiting time) in web server performance. In order to keep the response time within a satisfactory limit, web server maintainers often limit the number of concurrent open connections. When the number of requests overtakes this limit, all later access requests will be denied. Hence, often either the response time is long, or the user request is not processed at all. Either situation could lead to dissatisfied customers, driving many of them away from the site. In addition, the traffic may change with the time of the day, the day of the week, and even the month of the year. These seasonal or periodic fluctuations make it even more laborious to plan what would be an adequate capacity for a site.

3.3 Performance Management

The traditional performance management and measurement tools of the web server system are discussed from a system level point of view. A system level performance model portrays the system as a black box. In this case, the internal details of the box are not modelled explicitly. As a result, the throughput of the whole system is considered.

3.3.1 Performance Related Terminology

The concept of performance management includes quite many subsystems, like performance measurement, performance analysis, performance evaluation, etc. However, those concepts are not clearly defined, and they are inconsistently used in the literature.

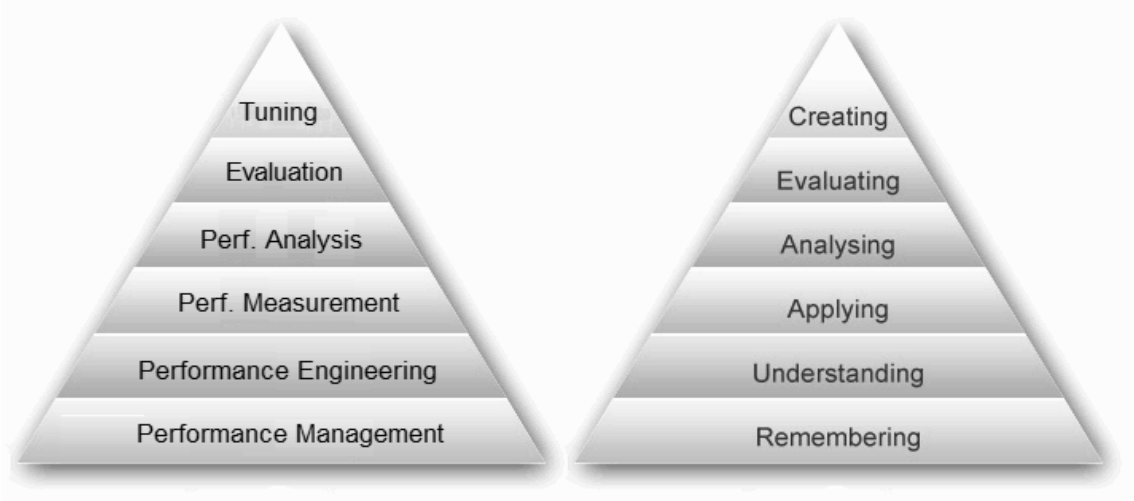

Figure 3.1: Performance management and its subsystems according to revised Bloom's taxonomy

Bloom's Taxonomy (Bloom, 1956; Buckley & Exton, 2003; Overbaugh, n.d.) helps define the concepts of performance measurement, performance analysis, and performance evaluation in relation to each other. The result is illustrated in Figure 3.1. The distinctive features of this approach are as follows (Singleton, 2002):

- An *automated process* in performance management aims to take the human "out of the loop". The human consideration is still required in order to set up the system and to define the performance goals. However, the mechanisms for observing the current state of the system, comparing the obtained values, deciding on tuning activities, and finally putting these activities into effect are automated and embedded within the system.
- Performance management accompanies a *distributed approach* to prevent any performance threat from a centralised coordinator, for it locally monitors system performance and gathers local state information.
- Performance management must include a mechanism that *autonomously controls* the system behaviour.
- Performance management should be *proactive* due to the time gap between a launch of control actions and state information collection. A prediction mechanism forecasts the following system state based on which control decision is made.

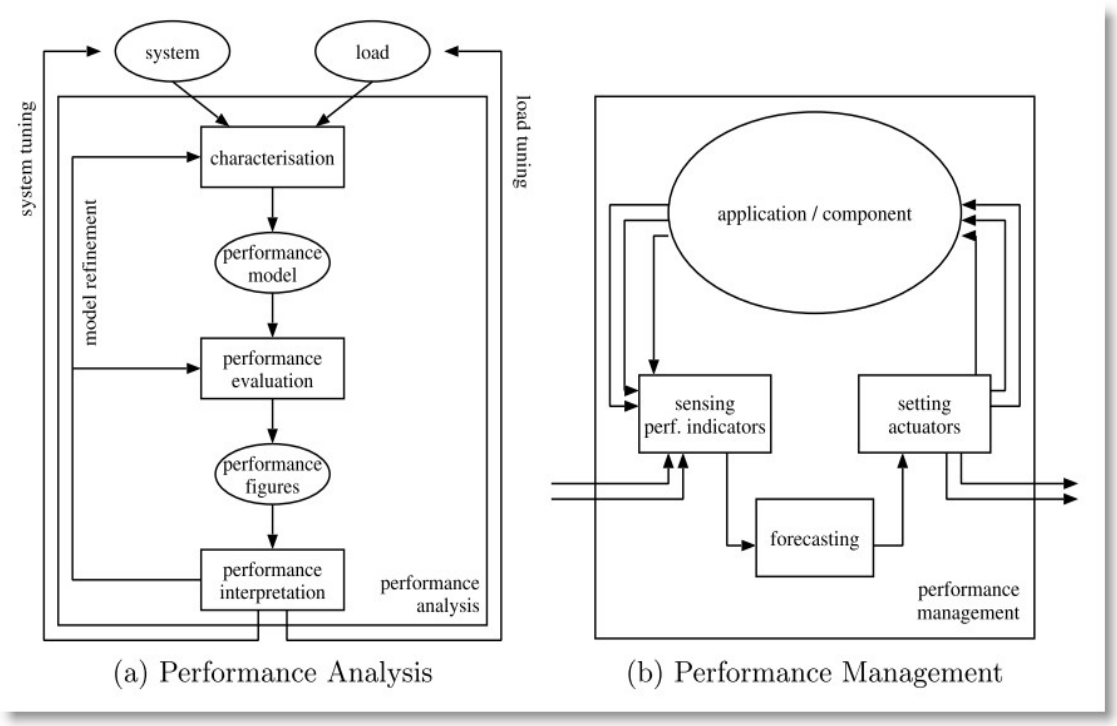

Figure 3.2: The traditional approach to evaluate the performance of computer systems and networks is an "off-line" performance analysis: (a) and an agent-based mechanism for the realisation of the performance management architecture; (b) (Kotsis, 2004)

A demand is observed for immediate embedded performance tuning actions replacing the conventional off-line approach of performance analysis. The human professional powered process of constructing a model, evaluating it, validating and interpreting the results and conclusion, thereby putting performance tuning actions into effect is no longer sufficient if real-time responsiveness of the system is needed. Hence, the proposition of an online, dynamic performance management approach which is outlined in Figure 3.2(b).

Figure 3.3 shows Performance Engineering activities and also as an example of the placement on waterfall development cycle (Singleton, 2002). It can be seen that the performance estimation and prediction can start as early as the system design. As soon as the first design has taken shape, the performance model construction can begin.

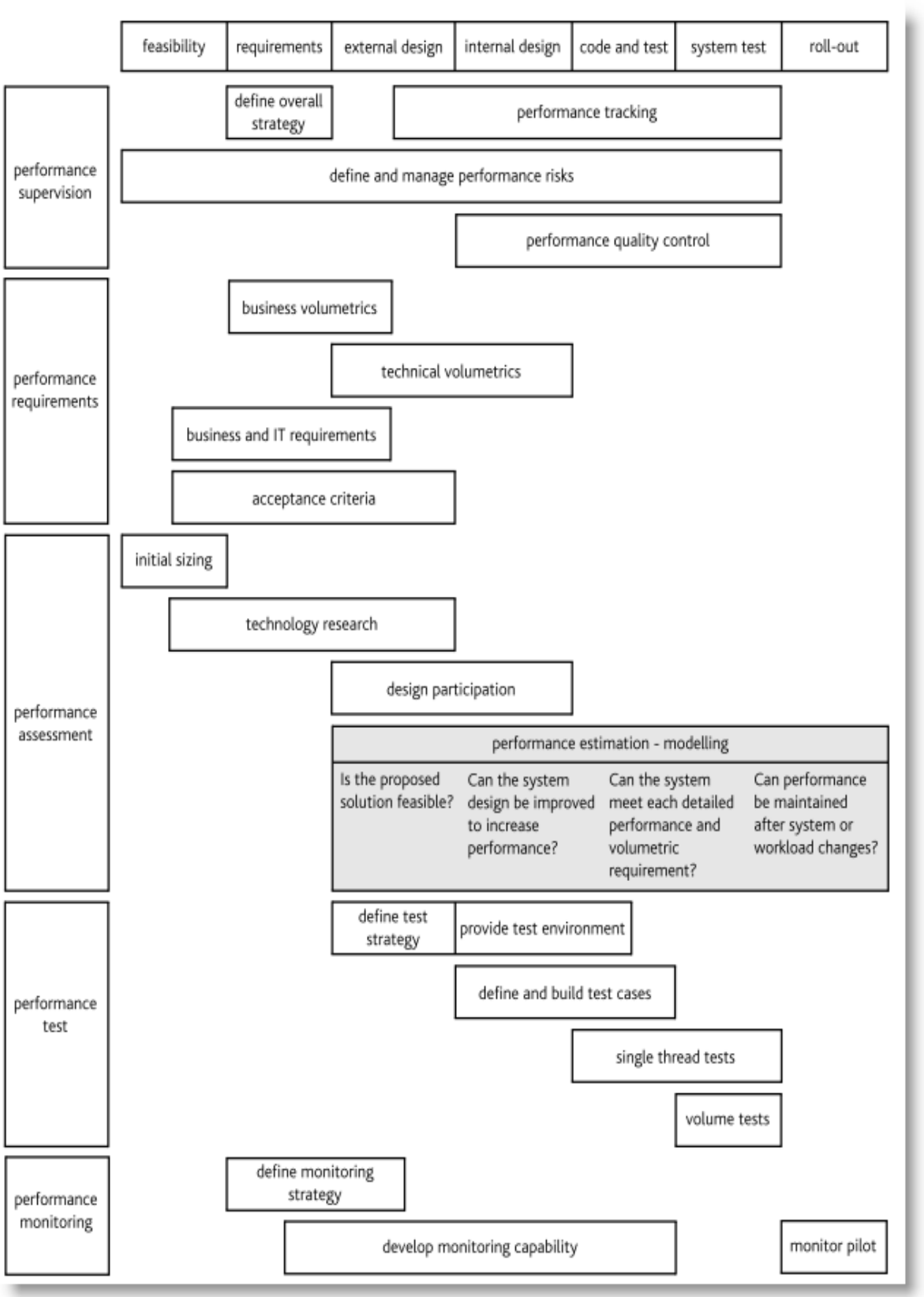

Figure 3.3: Modelling within performance engineering.

In practice, the application-induced load on the server in general is described, as well as the dimension of its workload. The workload is a set of variables and their values, which describe a variety of customer volumes and the volume of application activity over time. For example, in an online bookstore service, two descriptive parameters of the workload could be: (a) requests for the travel book purchase of a day; and (b) the web server CPU utilization per minute.

The term "workload" is used loosely to describe a general behaviour of client-server application. When the number of requests from customers does not vary considerably, server load is said to be stable. In contrast, when the request type or frequency changes with time, the server is said to be varying or dynamically loading. In addition, it is also used as a term referring to the amount of work required to provide the client resource requirements, which relate to the server.

3.3.2 Performance Engineering

In Jewell (2008), performance engineering is defined specially in software engineering as a technical discipline which aims to ensure that development project results in the delivery of a system which meets a pre-specified set of performance objectives. This is done by:

- managing the performance risk of a project
- controlling or coordinating activities in the project that have an impact on performance, and
- applying specialized performance estimation and design skills to the architecture of the system under development.

More generally, performance engineering is defined in Dumke et al. (2001) as a collection of models in support of the development of performance-oriented systems throughout the entire life cycle. Those models can be seen as a group of tests and are categorised and described in Splaine & Jaskiel (2001) as follows:

- *Smoke test* is used to evaluate whether the software release is ready for testing.

- *Load testing* is used to model the real-world performance of a website over a short period of time.
- *Stress test* is used to determine if a specific combination of hardware and software has the capacity to handle an excessively large number of transactions during peak operation hours.
- *Spike and bounce testing* is used to estimate the consequences of significantly exceeding a normal average amount of clients.

This categorisation is focused on the design and building phase of the web service. There are no tests for the whole life cycle of the web service. Jian (1991) recognized that performance evaluation is required at every stage in the life cycle of a computer system, including design, use, and upgrade. However, the presented analytical tools are improper to monitor web server system performance and capacity issues as a normal maintenance operation.

The scope of performance testing is not properly defined and varies from designing scripts and executing tests in its narrowest interpretation to all kinds of performance-related actions when a synthetic or natural workload is applied to the system in a much wider interpretation. The wider interpretation often includes performance analysis, performance troubleshooting and diagnostics, tuning, and capacity planning, making out the word "testing" to be rather concealing the substance of what is behind. The difference behind these definitions can partly describe the broad range of opinions about a path from functional testing to performance testing: while the path from automated functional testing to "narrow definition" performance testing is quite straightforward, skills required by other areas of a "wide definition" performance testing are quite different.

Speaking about the wide definition of performance testing, it is clearly and highly interlinked with performance engineering. Performance testing is a source of raw data for any kind of modelling and capacity planning activities. It is a way to calibrate and validate models. In this context, models mean not only formal models, but any kind of perception of how the system is supposed to work. If any proactive performance experiment is considered as performance testing, the only other source of data may be observation or log analysis of real work with the

system, and one may never know the exact level of load applied and what exactly happened in the system. A notable exception is data defining load (like throughput) which is a parameter of performance testing; so they cannot come from its results but should come from analysing real systems or other sources.

Performance is usually defined as the speed with which a certain operation is executed or the capability of executing a number of such operations within a unit of time. Performance analysis can be evaluated with simulation, analytical modelling or empirical evaluation (Koziolek, 2008; Lilja, 2000) as explained below:

- Simulation is an imitation of a programme execution focusing on specific aspects. It is flexible as changes can be dealt with easily if the simulation is derived automatically. However, simulation can suffer from a lack of accuracy.
- Analytical modelling is a technique where a system is mathematically described. Results can be less accurate than real-system measurements.
- Empirical evaluation is performed by measurements and metrics calculation. It provides the most accurate results since no abstractions are made.

Many websites are utilising multi-tier software architectures. The performance on such multi-tier environments is typically measured by the end-to-end response times. Most of the studies on modelling the response times have limited their focus to modelling the mean (Bhulai, Sivasubramanian, van der Mei, & van Steen, 2007). However, since the user experienced performance is highly affected by the variability in response times, the variance of the response times is important as well.

The increased complexity of web-based applications requires more server capacity. As a result, the experienced delay in loading the page is determined not simply by transfer delay but also by server performance. Normally, generating a page also involves connections to back-end mainframes or database servers, thereby slowing down the process even further.

Capacity planning is the process of predicting when future load levels will saturate the system and determining the most cost-effective way of delaying system saturation as much as possible. The lack of proactive and continuous capacity planning procedure may lead to unexpected unavailability and performance problems.

Capacity planning is essential for several reasons (Menascé & Almeida 2002):

- to avoid financial losses
- to ensure customer satisfaction
- to preserve a company's external image
- capacity problems cannot be solved instantaneously

Predicting the resource requirements in a rapidly changing environment brings more challenges, such as breaking news in the media environment or closing dates in e-government applications. In such a case, the normal load can be quite stable, but it can be increased rapidly, affecting system performance. The service provider cannot overestimate the performance of the system to keep the investments effective. From the service provider's viewpoint, this means seeking a balance between economically effective investments and customer satisfaction with the technical quality of the service.

3.3.3 Performance Estimation

The traditional approach to evaluating the performance of computer systems is an off-line performance analysis as represented in Figure 3.2(a). Originating from a characterisation of the system under study and characterisation of the actual load, a performance model is built, and performance results are obtained by applying performance evaluation techniques such as analytical, numerical, or simulation. Alternatively, the modelling and evaluation step can be replaced by performance measurements of the real system. In any case, an analysis of results follows, which can trigger an improvement of the model if the results do not provide the required

knowledge. Alternatively, it will lead to performance tuning activities if performance shortages are detected. However, many aspects of emerging computing environments create a variety of performance-results influencing factors, which cannot be adequately represented in a model. Applications in such environments are typically characterised by a complex, irregular, data-dependent execution behaviour, which is considerably dynamic and has time varying resource demands. The performance produced by the execution platform is laborious to predict as it is generally composed of heterogeneous components.

Individual servers or entire server system performance analysis and forecasting have been developed in a number of different models. Thalheim & Tropmann (2011) present a model where automatic performance analysis is difficult. A study by Tsai et al. (2007) in turn presents a model which assumes that only the system processing speed is the bottleneck.

One of the more important aspects of performance engineering is the selection of the appropriate estimating technique for the situation at hand. Figure 3.4 shows some of the common estimating techniques and the corresponding cost versus accuracy trade-offs. From a technical standpoint, the difference in these techniques comes from the manner in which volumetric and parametric costs, the resource model and queuing model are represented and used to estimate performance.

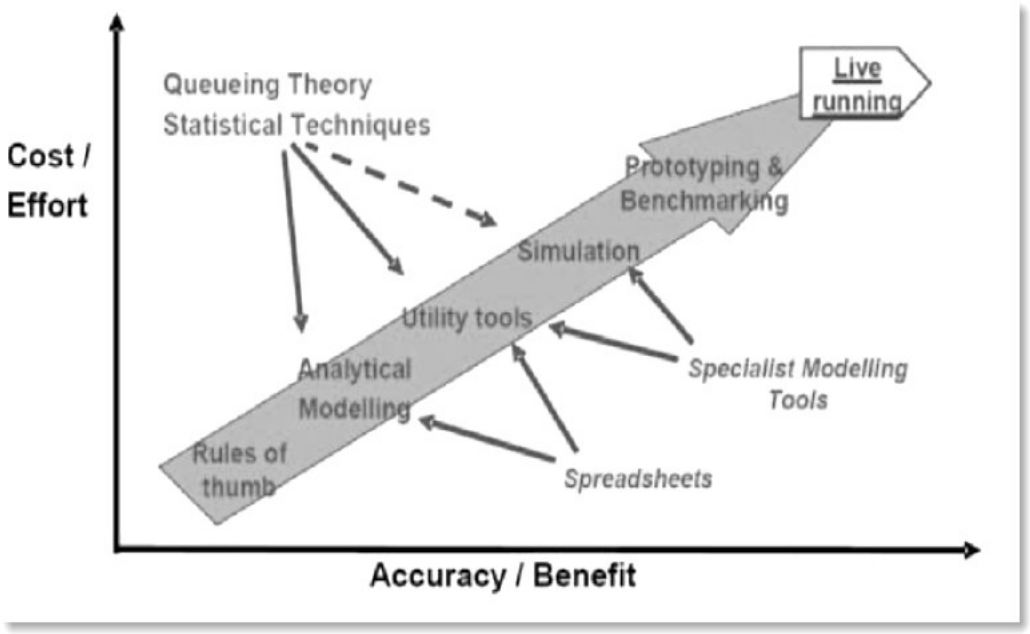

Figure 3.4: Cost/effort and accuracy/benefit trade-offs between performance estimation techniques (Jewell 2008)

Typically, more than one of these estimation techniques should be used during the life of a project. Low-cost, low-effort methods are commonly used early in the project for feasibility study purposes, whereas the higher-accuracy methods may be employed later, when more is known about the solution design and validation of the solution performance characteristics is deemed critical. These estimating techniques may be summarized as follows (Jewell, 2008):

- "Rules of thumb" estimating relies on very preliminary, simplified assumptions concerning volumetric and parametric costs, system resources and wait times in order to deliver an estimate relatively quickly.
- Analytical modelling uses spreadsheets (or special purpose tools in some cases) that perform static calculations to make predictions of "steady state" performance and utilisation of a system under a given workload. The static calculations typically take into account some amount of queuing theory and statistical techniques, in addition to volumetric, parametric costs and system resources.

- Simulation modelling conventionally employs what is known as a discrete event simulation tool to mimic the transaction processing behaviour of the live system from a resource utilisation and timing perspective. With these kinds of tools, the modeller must spend a considerable amount of time populating the model with the resource model, the parametric costs and transaction processing behaviour.

- Prototypes can be thought of as not entirely constructed or as early versions of a considered invention, product or solution. The purpose of building any prototype is to learn from and leverage the experience of building or testing such a prototype before making a commitment to the production version. For IT solutions, performance prototyping can serve as a means of investigating aspects of solution performance before a completely developed live system is available for performance testing.

- Benchmark testing refers to the process of performance testing using a known workload before and after making a change or enhancement to the system, in order to determine whether the system's performance has been impacted by the change.

As the previous categorisation shows, performance monitoring and estimation in a live environment are not known in general as an essential part of maintenance operations.

3.3.4 Performance Analysis

In Lilja (2000), performance analysis is described as a combination of *measurement*, *interpretation* and *communication* of a computer system's capacity. It means it is important to recognise that we need not necessarily be dealing with the complete system. Eventually, it is necessary to analyse only a small portion of the system, independent of the other components. For instance, we may be interested in studying the performance of a certain computer system's network interface independent of the size of its memory or the type of processor. Unfortunately, the components of a computer system can interact in incredibly

complex and frequently unpredictable ways. One of the most engaging tasks of the performance analyst can be figuring out how to measure the necessary data. A large quantity of creativity may be needed to develop good measurement techniques that disturb the system as little as possible while providing authentic reproducible results. After the required data have been collected, the results must be interpreted using appropriate statistical techniques. Finally, even excellent measurements interpreted in a statistically appropriate fashion are of no practical use to anyone, unless they are communicated in a clear and consistent manner.

Little's law is widely applicable in analytic methods. The law was first proven by Little (1961) and it applies insofar as the number of jobs entering the system is equal to those completing services, so that no new jobs are created in the system and no jobs are lost inside the system. The law can be used for a system or any part of the system. We can apply the law to relate queue length Q_i and response time R_i at the *i*th device as follows:

$$Q_i = \lambda_i R_i \tag{3}$$

where λ_i is the arrival rate to device *i*. In case of balanced job flow, the arrival rate is equal to the throughput X_i, equation (3) can be written thus:

$$Q_i = X_i R_i \tag{4}$$

Because Little's law can be used for a whole system, the equation (4) can be written for the whole system:

$$Q = X R \tag{5}$$

The response time at the maximum throughput is too high to be acceptable in many cases. In such cases, it is more interesting to know the maximum throughput achievable without exceeding a pre-specified response time limit. This may be called usable capacity of the system. In many applications, the knee point of the

throughput curve or the response time curve is considered the optimal operation point.

In any capacity analysis identifying the bottleneck resource should be the first step in performance improvement (Jian, 1991). The resource (e.g. memory, processor time, disk I/O, network I/O) with the highest total service demand D_i has the highest utilization and is called the bottleneck resource. Suppose the resource *b,* D_b is the bottleneck, D_{max} in a set of resources $D_1, D_2, \cdots, D_M$. The throughput bound of the system is:

$$X(N) \leqslant min\left\{\frac{1}{D_{max}}, \frac{N}{D+Z}\right\} \tag{6}$$

and the response time is respectively:

$$R(N) \geqslant max\left\{D, ND_{max} - Z\right\} \tag{7}$$

where *N* is the number of requests, *Z* is the think time, and $D = \sum D_i$ is the sum total of service demands on all resources.

Requests arrive at the web server at rate λ and they get served constantly at rate μ. The server utilization is $U = \lambda/\mu$. In the case of infinite request queue Menascé & Almeida (2002) has shown that the average response time *R* depends on average service rate μ and the utilization *U* of the server.

$$R = \frac{\frac{1}{\mu}}{1-U} \tag{8}$$

While the utilization increases close to 1, the denominator of Equation (8) goes to zero and R goes to infinity. In Figure 3.5, the curve shows a dramatic increase in response time seen when the workload approaches its maximum possible value.

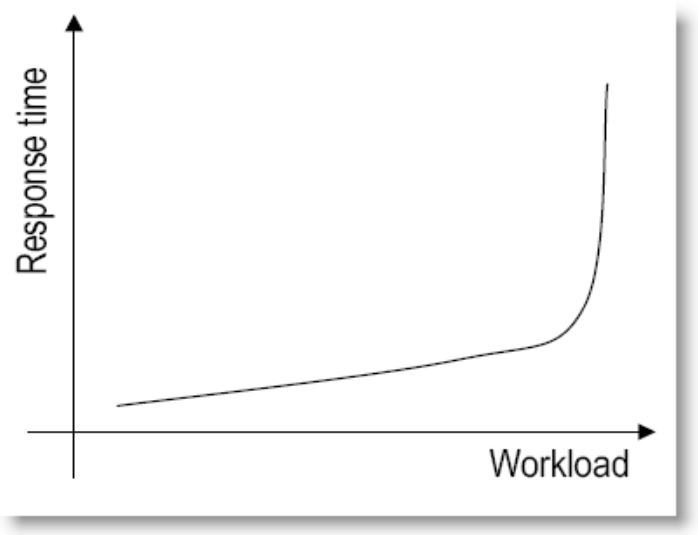

Figure 3.5: Typical website response time curve compared to workload

The arrival rate λ and the service rate μ may depend on the system state *k* in many cases. The throughput of the service system is usually a function of the number of requests present in the system. A typical throughput curve *X(k)* is shown in Figure 3.6. The light load region in the figure shows, that as the workload of website increases, the throughput increases almost linearly. At light loads, requests face very little congestion for resources. After some point, congestion starts to build up, and throughput increases at a much lower rate until it reaches a saturation value. The maximum value is determined by the bottleneck device at the single server.

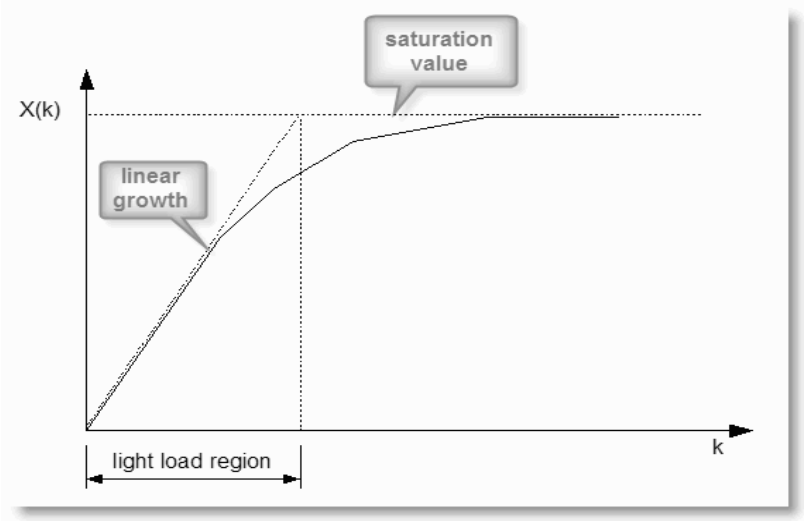

Figure 3.6: Typical website throughput curve compared to number of arrival requests (Menascé & Almeida, 2002)

Let J be the value of k, and after which the value of the throughput no longer changes. Now, the expressions for μ_k becomes:

$$\mu_k = \begin{cases} X(k) & , k \leqslant J \\ X(J) & , k > J \end{cases} \tag{9}$$

In general web systems, the user population size is infinite, and the queue size is limited to W requests. In case of $\lambda < X(J)$, the fraction of time, p_k, server has k $(k = 0,1,\cdots)$ requests (Menascé & Almeida, 2002):

$$p_k = \begin{cases} p_0 \dfrac{\lambda^k}{\beta(k)} & , k \leqslant J \\ p_0 X(J)^J \dfrac{\rho^k}{\beta(J)} & , k > J \end{cases} \tag{10}$$

where p_0 is:

$$p_0 = \left[1 + \sum_{k=1}^{J} \frac{\lambda^k}{\beta(k)} + \frac{\lambda^J}{\beta(J)} \times \frac{\rho}{1-\rho} \right]^{-1} \tag{11}$$

where $\beta(k)=X(1)\times X(2)\times\cdots\times X_k$ and $\rho=\lambda/X(J)$. As discussed in Section 2.5, the throughput function $X(J)$ depends on the internal state of the system and the rejuvenation interval.

3.4 Response Time

Response time is broadly defined as the time interval between a user's request for service and the service's return of results, as discussed in Fortier & Michel (2003). In reality, this is oversimplified and is not all there is to it. There are more elements on the side of both request and response that make up the true measure. The process begins with the user inputting the transaction. This is not a single step, but it can be much longer if the user is using an interactive interface with the transactional service. The database system must set up the appropriate data structures and provide resources for the transaction to be executed. The transaction is then executed by the database engine. The transaction then completes processing, prepares the transaction results and sends them off, as shown in Figure 3.7. Each of these steps, while a bit more complete than the simplistic model, is still only a partial representation of the full transaction processing cycle in a commercial database system. In addition, the rendering time required by the browser or any other end-user systems requires time that is dependent on the complexity of the data and the performance of the browser. Each of these components of the transaction response time is a response time component. These components are the subparts of the total transaction response time, just as queue wait time and server time represent the job time in a queuing model.

Selvidge et al. (2002) has defined *system response time* as the speed at which a computer responds to a user's command. User *think time* is defined as the duration of time between the computer's response and the user's next command input. For example, the time it takes from the user's action of selecting a link on a web page until the page is presented is the system response time.

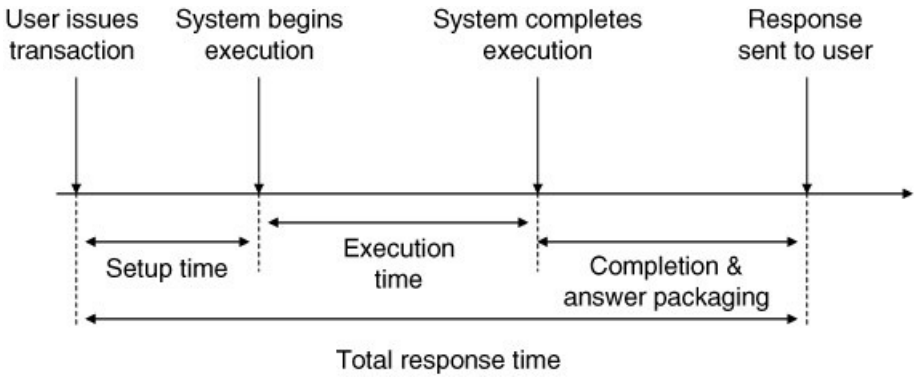

Figure 3.7: Transaction processing response partitioning

The response time of a computer system will normally rise as the load increases. Manifold methods have been developed to provide rules of thumb for such scenarios. One, called the stretch factor (Fortier & Michel, 2003), is computed as the expected response time over the expected service time, or:

$$Stretch\ factor = \frac{E_W}{E_S} \tag{12}$$

where E_W is the expected response time and E_S is the expected service time. This measure is depicted in Figure 3.8. In most real systems, one wishes to see this stretch factor have a computed value of approximately 5. If the factor rises above this approximation, this implies longer waiting times in relation to service times and therefore, lower availability of the resource and higher utilization.

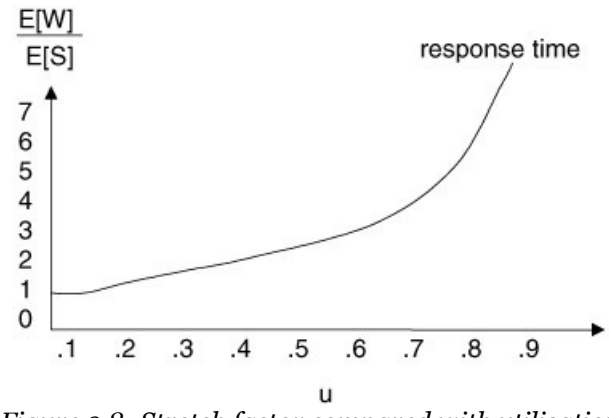

Figure 3.8: Stretch factor compared with utilisation

The ISO 9126-2 (ISO/IEC, 2001) standard defines three different response times. In general, the response time should measure the time consumption for completing at specified task. It should be recorded as the *time span between the start of the task and its completion.*

The second definition is the *mean time to response*: The meantime to response should record the regular response time under a defined system load in terms of concurrent tasks and system utilization. It is estimated by acquiring the response several times and dividing the sum of all response times by the number of measurements. This can again be divided into the required mean response time so that the result is the ratio of fulfilling the prerequisites. The ratio should be less than 1.0, lower being better.

Finally, the *worst case response time* is defined: The worst case response time is calculated using the ratio of the maximum response time of a set of measurements divided by the required maximum response time. Again, the value should be less than 1.0, lower being better.

3.5 Throughput

Throughput is defined as the rate (request per unit of time) at which the requests can be serviced by the system. For batch streams, the throughput is measured in jobs per second. For interactive systems, the throughput is measured in request per second. For CPUs, the throughput is measured in Millions of Instructions per Second (MIPS), or Millions of Floating-Point Operations per Second (MFLOPS). For web servers, the throughput is measured in HTTP operations per second (HTTPops/sec). For networks, the throughput is measured in packets per second (pps) or bits per second (bps). For transactions processing systems, the throughput is measured in Transactions per Second (TPS). Hence, in order for the throughput value to be meaningful, the type of the transaction considered has to be characterised when reporting the throughput.

Throughput is defined in some studies (Fortier & Michel, 2003; Jian, 1991; Koziolek & Happe, 2008) as with response time, which will grow as additional load

is placed on the system. However, unlike response time, there will be a point when the throughput will maximize and possibly begin to degrade, as shown in Figure 3.9. In the figure, the throughput increases over a wide range of load and then slows as a saturation point is reached. In the throughput case, the throughput increases to some maximal level and then levels off. At a critical point in the load, where the response time has begun to increase exponentially, the throughput begins to degrade below the maximum. Such curves are typical of computer systems where there is inadequate service capacity for the presented load. It is aimed always at keeping the throughput near its peak, but not too far into the saturation region, so that resources stay available for spikes in load. The maximum achievable throughput under ideal workload conditions is called the nominal capacity of the system. For computer networks, the nominal capacity is called the bandwidth and is usually expressed as bits per second. Often the response time at maximum throughput is too high to be acceptable. In such cases, it is more relevant to know the maximum throughout achievable without exceeding a pre-specified response time limit. This may be called the usable capacity of the system. In many applications, the knee of the throughput or the response time curve is considered the optimal operation point.

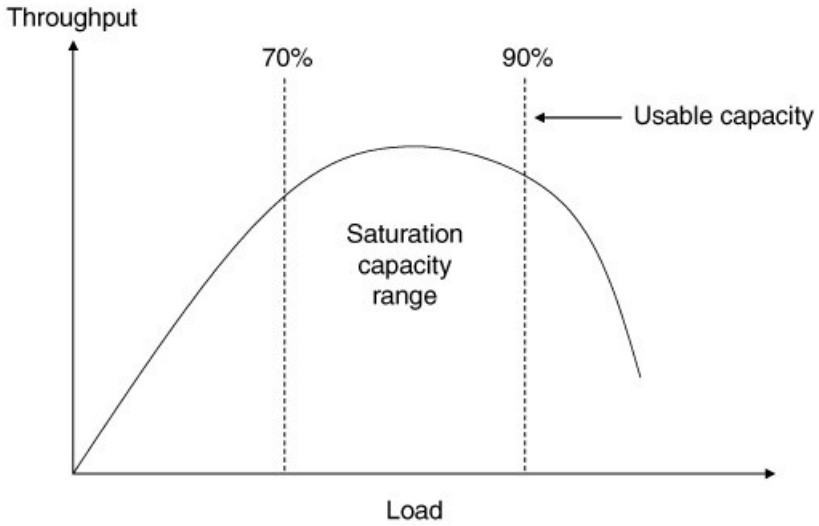

Figure 3.9: Throughput curves versus response curves.

Another important measure is efficiency. This measure is related to utilization and throughput. The relationships look at the ratio of the maximum achievable throughput compared to the actual throughput:

$$Efficiency = \frac{real\ throughput}{theoretical\ throughput} \tag{13}$$

Efficiency can also be measured for multiple resource systems. One common use is when looking at the performance speed-up of having one processor versus n processors. Efficiency in this class of environment is calculated as the ratio of the theoretical throughput times the number of devices, and divided by the speed of a single device.

In Figure 3.10 we can see that the theoretical efficiency of adding more processors is a linear curve with an efficiency equal to the number of devices applied. The real measured curve shows a very different story. The efficiency is not linear and continues to degrade as more devices are added. This is due to the added overhead involved in keeping the processors effectively utilized in performing tasks.

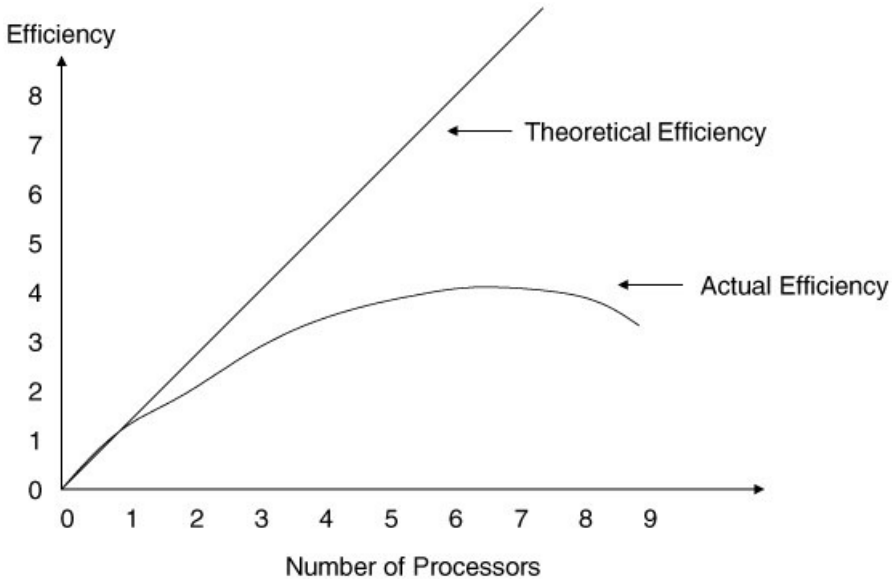

Figure 3.10: Multiprocessor efficiency curve.

Throughput is defined in ISO 9126-2 (ISO/IEC, 2001) as follows: The *throughput* characterises the number of tasks, which can be satisfied over a given period. In

addition, the *mean amount of throughput* is described as a number of concurrent runs in the specified task calculated by the sum of each of the throughputs and divided by the amount of runs. Then, this is divided by the required throughput to get a ratio. The ratio should be less than 1.0, lower being better. Third, the *worst case throughput* is defined as the amount of concurrent runs in the specified task, to be calculated by taking the maximum of the measured throughput values and dividing this by the required throughput to get the ratio. The ratio should be less than 1.0, lower being better.

3.6 Utilization, Reliability, and Availability

The utilization of a resource is a measure of how busy the resource is (Jian 1991; Fortier & Michel 2003). It is measured as the fraction of time the resource is busy servicing the requests. Thus this is the ratio of busy time and total elapsed time over a given period. The period during which a resource is not being used is called the idle time. Some resources, such as processors, are always either busy or idle, so their utilisation in terms of the ratio of busy time to total time makes sense. For other resources, such as memory, only a fraction of the resource may be used at a given time; their utilisation is measured as the average fraction used over an interval. It is computed as the fraction of time the resource is busy servicing clients divided by the entire period:

$$Utilization = \frac{time\ \ busy}{(time\ \ busy + time\ idle)} \tag{14}$$

In most systems, it is not reasonable to saturate resources. Instead, the aim is to balance the utilization such that no device is more heavily utilized than another. In principle, this is the goal, but in reality, this is difficult to achieve. Utilization is an important measure when examining systems. Different devices in a system have different average utilization values. For example, processors typically will be highly utilized, while memory, disks, and other peripheral devices will all have smaller fractional use time.

Utilisation is defined in ISO 9126-2 (ISO/IEC, 2001) as having three purposes: The *I/O device utilisation* section suggests several metrics to describe the load of the specified resources with respect to the tasks defined. It contains metrics for the device utilisation, load limit, I/O related errors and the waiting time of the user due to device response times. The *memory resource utilisation* metrics can be used to conclude the memory consumption for the execution time of the specified tasks. The standardised metrics include metrics for the maximum amount of memory consumed, the mean occurrence of memory errors, and the ratio of memory errors to execution time. Finally, the *communication resource utilisation* is supposed to characterize the load of communication-related transmission channels. Metrics of this group contain the maximum transmission utilisation, the media device utilisation balancing, the mean occurrence of transmission errors, and the mean of transmission errors per time.

Other important measures in analysing computer systems include systems reliability and systems availability. They are defined in (Jian, 1991). The reliability of a system is a measure of the probability of errors or a measure of the typical time between errors. Most computer systems are fairly reliable, with hardware being more reliable than software. The availability of a system is measured in terms of reliability. If a system is highly reliable, it will more likely be available than not. However, if a system is unreliable, then it will have periods of downtime, where the system is not running or is running erroneously. The time during which the system is not available is called downtime; the time during which the system is available is called uptime. Often, the mean uptime, better known as the Mean Time to Failure (MTTF), is a better indicator since a small downtime and small uptime combination may result in a rather high availability measure, but users may not be able to get any service if the uptime is less than the time required to complete the service.

3.7 Benchmarking Tools and Techniques

Fortier & Michel (2003) described four techniques for computer system performance evaluation including simulation modelling, Petri-nets, analytical

modelling, and test-bed analysis. Depending on the criteria allocated for the computer system's analysis, some approximate selection metrics can be determined. The most significant criterion deals with this stage of the computer system's life cycle. For example, if the computer system planning is in the earliest phases of the life cycle, when trade-offs on new components are examined, the analytical modelling is the most effective method to provide relatively quick answers, making it possible to determine early on if a subset of several alternatives is best for a more detailed modelling. Once this rough analysis has been completed, and choices of alternatives are narrowed to some smaller subset, Petri-nets would probably be applied to further refine the choices. Petri-nets add the ability to model and trade off concurrency, conflict, and synchronization, something that is impossible to accomplish with analytical modelling. In the next phase, when the system or something similar already exists, measurements are available as a modelling possibility. Simulation provides the ability to produce detailed models of a target system or just some specific contentious component(s). Once the system is constructed, the empirical modelling would be applied. This would allow for verifying whether early modelling was correct and to possibly identify areas where the new system could be further refined and improved.

The next criterion for consideration when deciding on which modelling tool to use is the resources that have to do with the modelling task. In most situations, a model is requested after some problem has occurred, and it should be resolved as soon as possible. If time is not bounded to perform all the possible evaluations, then each reasonable model would probably be walked through, thus refining the analysis defined under the criterion of the time stage. The problem is that typically there is no such luxury available. If time is bounded, then the use of analytical or Petri-net modelling is the most effective, with analytical modelling winning out if time is very short. If time is important though not critical, then Petri-nets and simulation are the next models of choice. Petri-nets require less time to develop than simulations but would also be provided with possibly less detailed analytical information. If the system already exists, then measurements may be appropriate over simulation modelling, if the number of alternatives we are looking at is small.

If the number of alternatives is considerable, then simulation would win out, even though it would typically take more time than measurements.

Two remarkable associations have developed multi-purpose load test applications since the early days of commercial computing, SPEC (*Standard Performance Evaluation Corporation*) (SPEC, 2010) and TPC (*Transaction Processing Performance Council*) (TPC, 2010). Both organizations have benchmarking tools for several types of transactions on internet computing. However, both tools use their own generic applications on the application server. They do not measure the real existing application set. The result describes the performance of the hardware, neither the configuration nor the application. Especially, the actual usage and its change on time do not affect the results. As we have discussed in Section 2.5, software aging is remarkably effective on the performance of the server system.

Producing representative web trace load is a challenging task due to the number of unusual features of web workload. Web servers have encountered highly variable requests, which are exhibited as variability in CPU loads and the number of open connections. The second feature of web workload is self-similarity in network traffic of web requests, i.e. traffic can show significant variability over a wide range of scales. Self-similarity in traffic has been shown to have a significant negative impact on network performance (Dill et al. 2002; Crovella & Bestavros 1997; Barford & Crovella 1998), so it is an important feature to capture in a natural workload.

To clarify these properties in a benchmarking process, one of the following approaches could be used: a trace-based or an analytic approach. Trace-based workload generation uses pre-recorded records of past workloads typically identified in access logs and either samples or replays traces to generate workloads. In contrast, analytic workload generation starts with mathematical models for different workload characteristics and then generates outputs that adhere to the models.

Both approaches have strengths and weaknesses. The trace-based approach is easy to implement, and it mimics the activity of a known server system. However, it treats the workload as a black box. Hence, the result is that the response to the

system is of well known workload only. Furthermore, it can be hard to adjust the workload to imitate future conditions or varying demands. Analytic models do not have these weaknesses, but they can be challenging to construct for at least four reasons. First, it is necessary to identify those characteristics of the workloads which are important to model. Second, the chosen characteristics must be empirically measured. Third, it can be difficult to create a single output workload that accurately exhibits a large number of different characteristics. Fourth, updating the model is required in case of application, configuration or hardware updates, and a lot of laborious operations may be required.

Benchmarking tools and techniques are reviewed in a tutorial (Andreolini et al. 2002). It is aimed at evaluating the performance and scalability of highly accessed web server systems. The focus in the study is on design and testing of locally and geographically distributed architectures where the performance evaluation is procured through workload generators and analyses in a laboratory environment. The tutorial perceives the qualities and issues of existing tools in terms of the main features that characterize a benchmarking tool (workload representation, load generation, data collection, output analysis and report) and their applicability to the analysis of distributed web server systems.

One study (Ruffo, Schifanella, Sereno, & Politi, 2004) presents a set of tools that allows the performance analysis of web applications by means of a scalable what-if analysis on the test bed. The approach used in that paper is based on a workload characterization generated from information extracted from log files. The workload is generated using user behaviour analysis, which is derived by extracting information from the web application log files. In this manner, the synthetic workload used to evaluate the web application under test is representative of the real traffic that the web application has to serve. One of the most common critics to this approach is that synthetic workload produced by web stressing tools is far from being realistic. The use of the behaviour analysis might be useful to overcome this criticism.

Grid computing is a common platform for solving large-scale computing task requirement for high availability. However, a number of major technical issues, including the lack of sufficient performance evaluation approaches, disturb the

further development of grid computing. Therefore, the requirements are manifold; adequate approaches must synthesize appropriate performance metrics, natural workload models, versatile tools for workload generation, submission, and analysis. An approach to intercepting this complex problem is shown in Iosup et al. (2007). A set of grid performance objectives, based on traditional and grid-specific performance metrics, is shown. Also, the requirements for realistic grid workload modelling, data and network management, and failure modelling are synthesized.

Common web server systems are relying on multi-tier architectures. The performance of such multi-tier systems is typically measured by the end-to-end response times. Most of the research studies analysing the response times of such systems have restricted their focus to modelling the mean. However, since the user-perceived performance is highly affected by the variability in response times, the variance of the response times is important as well. A study (Bhulai et al., 2007) has presented an analytical model for multi-tiered web applications based on a queuing-theoretical framework. Based on this model, the mean response time of a service can be estimated and a reasonable approximation of its variance can also be provided.

An end-to-end monitor to measure website performance has been developed by Cherkasova et al. (2002). The system passively collects packet traces from the server site to determine service performance characteristics. The study introduces a two-pass heuristic method and a statistical filtering mechanism to accurately reconstruct a composition of individual page and performance characteristics integrated across all client accesses. However, the monitor requires specific software components (agents) for each server of an independent network appliance in order to capture all HTTP transactions for a web server.

Ardaiz et al. (2001) have shown that with measurements taken from the server and without introducing traffic on the network, it is possible to estimate the service time experienced by a web client. In the study, the metrics has been analysed so as to be obtainable at a web server regarding the service time experienced by a client, which has components that depend on the round trip time and bandwidth, something that is difficult to obtain without modifications in all web browsers. This metrics has been compared in an experiment with clients and servers situated

in different locations on the Internet, and therefore, under the same conditions of variability and network load as any normal client would experience.

3.8 Monitoring Tools

One report (Balaton, Kacsuk, Podhorszki, & Vajda, 2000) provides a short description of the grid monitoring architecture and existing event services. The report compares several monitoring tools that can be utilised in a grid environment. Compared properties of the tools are based on the requirements of scalability, intrusiveness, validity of information, data format, extendibility, communication, security and measurement metrics. The study is aiming to collect monitoring data in large distributed systems for a variety of purposes such as performance analysis, tuning, and prediction. As a result, the study presents a list of required features for monitoring tools. Most of the compared tools are using their own, albeit quite well-known, proprietary agents to collect the raw data.

Distributed computer systems require a great amount of monitoring data to be collected for a variety of tasks such as fault detection, performance analysis, performance tuning, performance prediction, and scheduling. A study by Tierney et al. (2001) presents a developed agent-based system to automate the execution of monitoring sensors and the collection of event data. The developed Java Agents for Monitoring and Management (JAMM) architecture relies on well-known sensors in the UNIX world, like *netstat*, *iostat*, and *vmstat*. However, JAMM is planning to collect monitoring events, and to analyse and visualise them with external application like *NetLogger Toolkit* (Tierney et al., 1998). It can be used even as an accurate application level analysis and lifeline style visualization. In addition, there are no possibilities to integrate them into the access history or well-known load simulation.

Most of the analytic applications and monitoring systems have their own features and are not necessarily interoperable due to different designs and implementation approaches. The study by Yang & Theys (2005) presented a resource monitoring framework (RMF) that provides network administrators and researchers with a

consistent and scalable interface for a wide range of monitoring applications. However, this framework does not include tools for integrating the access logs and well-known load simulation.

Most of the tools mentioned above are relying on their own sensors for data collecting. However, in modern computer systems, robust, comparable, reliable, and secure sensors are integrated into the kernel level. The data of these sensors can be collected by using SNMPv2 or SNMPv3. Most server resources are rather simple to implement, like the memory or TCP connection available. At the same time, CPU usage is the most interesting in several service instances and its implementation is playing a critical role.

On the other hand, those tools analyse all the individual user requests and network packages, referring to certain requests. The concept leads to accurate results. However, in geographically distributed systems this has to be distributed to the sensors as well. For example, local area network (LAN) traffic cannot be remotely monitored without a local proxy. The local proxies require investments, maintenance, and these are increasing monitoring costs. The monitoring and analytic systems on this type of arrangement are poorly generalized with increased energy consumption.

The CPU is a discrete state machine. It can only be at 100%, executing an instruction, or at 0%, waiting for something to do. There is no such thing as using only 45% of a CPU. The CPU percentage is the amount of time interval that the system's processes were found to be active on the CPU. If a process is taking 45% CPU, 45% of the samples taken are active on the CPU. In the rest of the time, the process was in a waiting state. Load averages do not include any processes or threads waiting on I/O, networking, databases or anything else not demanding the CPU. Hence, it is precisely the CPU load that is measured. The load averages differ from CPU percentage in two significant ways: 1) load average measures the trend in CPU utilization, and not only an instantaneous snapshot, as does percentage; and 2) load averages include all demands for the CPU, and not only how much was active at the time of measurement. The CPU usage monitoring in Unix-liked operating systems is based on time counters in the kernel. Walker (2006) described the CPU load instrumentation in Linux kernel as follows.

In the kernel of Linux, each dispatchable process is granted a permanent amount of time on the CPU per dispatch. By default, this amount is 10 milliseconds. For such short time duration, the process is assigned a physical CPU on which to run its instructions and is allowed to take over that processor. In most cases, the process will give up control before the 10 ms are up through socket calls, I/O calls or calls back to the kernel. On a typical modern 2.6 GHz processor, 10 ms is enough time for approximately 50-million instructions to occur. If the process uses its fully allotted CPU time of 10 ms, an interrupt is raised by the hardware, and the kernel regains control of the process. The kernel then promptly penalizes the process for being such a hog. Hence, time slicing is an important design concept for making an operating system seem to run smoothly on the outside.

One of the duties the kernel completes when it receives control is to increase its jiffies counter. The jiffies counter is for measuring the number of time slices that have occurred since the system was booted. The jiffies counter is incremented by 1, and the load-average calculation is checked to see if it should be computed. In actuality, the load-average computation is not truly calculated on each quantum tick, but driven by a variable value that is based on the HZ frequency setting and tested on each quantum tick. HZ is not the processor's MHz rating. This variable sets the pulse rate of particular Linux kernel activity, and 1 HZ equals one quantum or 10 ms by default.

The data values can be figured out using a */proc* pseudo-filesystem as an interface to kernel data structures. In Linux and Solaris, it is mounted at */proc*. It carries a great number of information in kernel. Data for SNMP interface in Linux can be figured out using the next source code fragment on Linux and Solaris kernels.

```
read("/proc/stat", buff);
sscanf(buff, "%llu %llu %llu %llu %llu %llu %llu", &cusell,
&icell, &csysll, &idell, &iowll, &cirqll, &csoftll);

cpu->user_ticks = (unsigned long)cusell;
cpu->nice_ticks = (unsigned long)icell;
cpu->sys_ticks = (unsigned long)csysll;
cpu->idle_ticks = (unsigned long)idell;
cpu->wait_ticks = (unsigned long)iowll;
cpu->intrpt_ticks = (unsigned long)cirqll;
cpu->sirq_ticks = (unsigned long)csoftll;
```

The corresponding code fragment used in Solaris is the following:

```
kstat_read(kstat_fd, ksp, &cs);
cpu2->user_ticks = (unsigned long)cs.cpu_sysinfo.cpu[CPU_USER];
cpu2->idle_ticks = (unsigned long)cs.cpu_sysinfo.cpu[CPU_IDLE];
cpu2->kern_ticks = (unsigned long)cs.cpu_sysinfo.cpu[CPU_KERNEL];
cpu2->wait_ticks = (unsigned long)cs.cpu_sysinfo.cpu[CPU_WAIT];
/* or cs.cpu_sysinfo.wait[W_IO]+cs.cpu_sysinfo.wait[W_PIO] */
cpu2->sys2_ticks = (unsigned long)cpu2->kern_ticks+cpu2->wait_ticks;
/* nice_ticks, intrpt_ticks, sirq_ticks unused */
```

The above code fragments show that implementations are not the same even in different Unix versions. However, there are some common principles in most operating systems, i.e. the metrics of load is characterized as *user*, *nice*, *sys*, *idle*, and *wait* states. The names may vary in different environments. All these categories have their own counters. In addition, some implementations have the *raw* counter. The different counters are available via SNMP interface for monitoring purposes.

The single value of a number of ticks does not say anything about the load rate. It requires also knowledge about the duration of a time slice between two different values. This leads to the concept of load average. The load averages indicate by increasing duration whether (or not) the physical CPUs are over- or under-utilized. The point of perfect utilization, meaning that the CPUs are always busy and yet no process ever waits for one, is the average from matching the number of CPUs. If there are four CPUs on a machine and the reported one-minute load average is 4.00, the machine has utilized its processors perfectly for the last 60 seconds. This understanding can be extrapolated to the 5- and 15-minute averages.

There are plenty of SNMP implementations available in different operating systems and in different architectures. In this study, the widely available net-SNMP (Net-SNMP Development Team, 2007) is used in the Linux environment. In the MIB-description, the CPU usage is described as follows:

```
ssCpuRawSystem OBJECT-TYPE
  SYNTAX   Counter32
  MAX-ACCESS read-only
  STATUS   current
  DESCRIPTION
    "The number of 'ticks' (typically 1/100s) spent
     processing system-level code.

     On a multi-processor system, the 'ssCpuRaw*'
     counters are cumulative over all CPUs, so their
     sum will typically be N*100 (for N processors).

     This object may sometimes be implemented as the
     combination of the 'ssCpuRawWait(54)' and
     'ssCpuRawKernel(55)' counters, so care must be
     taken when summing the overall raw counters."
  ::= { systemStats 52 }
```

In net-SNMP, there are several indicators for CPU usage. Some of them indicate percentage values and the rest are for CPU ticks. The tick indicators are called 'Raw'. The total number of ticks is $100 * \textit{Number of Processors}$. For a single processor machine numbers will appear to be percentages as the kernel will tally ticks at 100 per second.

```
UCD-SNMP-MIB::ssCpuUser.0 = INTEGER: 7
UCD-SNMP-MIB::ssCpuSystem.0 = INTEGER: 0
UCD-SNMP-MIB::ssCpuIdle.0 = INTEGER: 91
UCD-SNMP-MIB::ssCpuRawUser.0 = Counter32: 105129968
UCD-SNMP-MIB::ssCpuRawNice.0 = Counter32: 5481391
UCD-SNMP-MIB::ssCpuRawSystem.0 = Counter32: 9448037
UCD-SNMP-MIB::ssCpuRawIdle.0 = Counter32: 932770472
UCD-SNMP-MIB::ssCpuRawWait.0 = Counter32: 1144770
UCD-SNMP-MIB::ssCpuRawKernel.0 = Counter32: 0
UCD-SNMP-MIB::ssCpuRawInterrupt.0 = Counter32: 1442
```

System CPU raw user (ssCpuRawUser): user CPU time in ticks per second, as reported by the kernel.

*System CPU raw nice (*ssCpuRawNice): nice CPU time in ticks per second, as reported by the kernel.

System CPU raw system (ssCpuRawSystem): system CPU time in ticks per second, as reported by the kernel.

System CPU raw idle (ssCpuRawIdle): idle CPU time in ticks per second, as reported by the kernel.

System CPU raw wait (ssCpuRawWait): iowait CPU time in ticks per second, as reported by the kernel.

System CPU raw kernel (ssCpuRawKernel): kernel CPU time in ticks per second, as reported by the kernel.

As described, the interface is suitable also for the multi-CPU architecture. The timing of each value has to be managed by the application as well as the number of CPU's that has to be known by the application.

In the Microsoft Windows environment, the corresponding CPU busy value is available, e.g. in SNMP-Informant MIB. The description of the MIB is as follows:

```
cpuPercentProcessorTime OBJECT-TYPE
 SYNTAX   Gauge32
 MAX-ACCESS read-only
 STATUS   current
 DESCRIPTION
     "% Processor Time is the percentage of time
     that the processor is executing a non-Idle
     thread. This counter was designed as a primary
     indicator of processor activity. It is
     calculated by measuring the time that the
     processor spends executing the thread of the
     Idle process in each sample interval, and
     subtracting that value from 100%. (Each
     processor has an Idle thread which consumes
     cycles when no other threads are ready to run).
     It can be viewed as the percentage of the
     sample interval spent doing useful work. This
     counter displays the average percentage of busy
     time observed during the sample interval. It
     is calculated by monitoring the time the
     service was inactive, and then subtracting that
     value from 100%."
  ::= { processorEntry 5 }
```

As we can see, the CPU load value in Linux is the number of ticks, and in Microsoft Windows it is the load-average. However, the sample interval is not defined explicitly. Empirical test has shown, that the results for SNMP queries in both environments are accurate compared to the load monitoring tools on each

operating system; similar results are found for the *top* in Linux and the *Task Manager → Performance* in Microsoft Windows.

All these performance measurements are not useful unless there is some relationship associated with the measure. For example, how do we know if, for some given metric, it is important to maximize or minimize its value? To make sound judgements, we must understand the measures we are taking and what their relationship is to system values, as shown in Figure 3.11. For example, for a CPU, do we wish to have a high number of instructions per second or a low number? Are we looking for medians or modes? This will make a difference on how the results will be interpreted. To decide on how to interpret the measurements, we must understand how they are related to each other. For example, high disc utilization may be mapped to a low system throughput. Alternatively, high CPU utilization may be mapped to a high throughput. It is important to know which is which in order to decide.

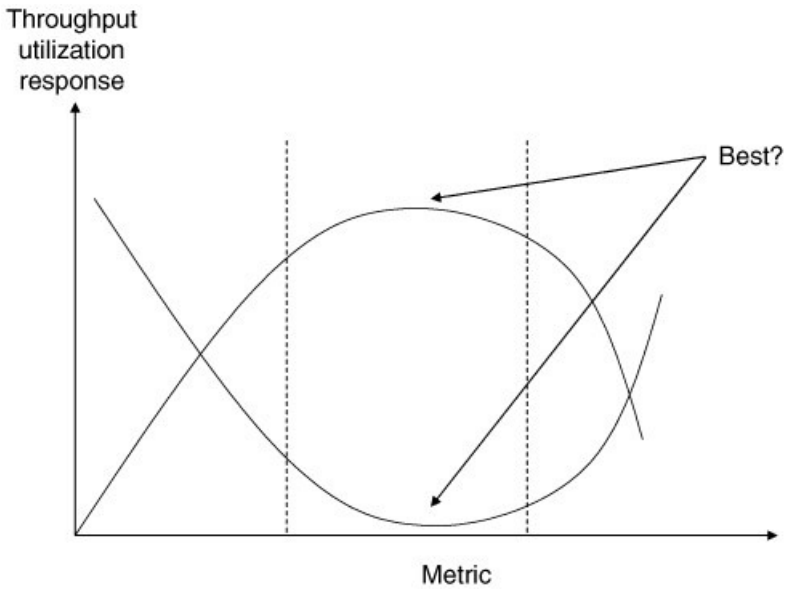

Figure 3.11: Metrics versus usefulness.

The load average from 0 to 1.0 are acceptable for a single CPU. As a general rule of thumb, a machine is being overworked if load averages consistently exceed three times the number of CPUs. However, the average number of processes is not the right criterion to evaluate the load of the server system. In a long period, the CPU

can be completely loaded and the number of queuing processes is close to zero if the arrival rate is equal to the processing time. At the same time, the load of CPU is close to 100%.

3.9 Performance Prediction Using Natural Load

Draheim et al. (2006) presents a load testing of web applications by simulating realistic user behaviour with stochastic analysis models. Simulation of user behaviour is necessary in order to achieve valid testing results. In contrast to many other user models, website navigation and time delay are modelled stochastically. The models can be constructed from sample data and can take into account the effects of session history on user behaviour and the existence of different categories of users. The approach is implemented in an existing architecture modelling and performance evaluation tool, and is integrated with existing methods for forward and reverse engineering. This is a typical agent-based performance tool suitable for in-house application development process. However, it does not suit the industry-strength web server application tools.

A key aspect in managing resources for customer sites is to predict and assess the load associated with a site in order to figure out how best to allocate resources for the site over time and to efficiently schedule tasks. The cost associated with the site, as well as return on investment, is also key parameters. A paper (Bagchi, Hung, Iyengar, Vogl, & Wadia, 2006) has described the work done in developing tools for answering these critical questions. The tools use both analytical models and discrete event simulations to predict performance and analyse costs needed for handling a customer workload while satisfying the service level objectives. These tools provide capacity and load planning, performance simulation, as well as cost and financial analyses.

A study by Hadharan et al. (2000) indicates that for the Active Server Page (ASP) Script Engine, performance predictions from the simulation model matched the performance observed in a test environment. However, for the JSP Script Engine, the model predicted a higher throughput than laboratory test results at high load.

This result suggests that web server performance can be critically limited by a software bottleneck that causes requests to be serialized. This may cause a request to wait for some resource (i.e., a lock) in contrast to consuming CPU or memory. In addition, the study ignores the limitations caused by inadequate configuration.

The study by Bacigalupo et al. (2004) reports on a comparative assessment of two approaches for predicting mean response times to different workloads on server architectures for a robust application benchmark. Results introduced show that both manners can be used to make predictions for new server architectures with a valid level of accuracy. The study also shows that the method can make close predictions when only a very restricted amount of recorded data is available. The work is extended to show that percentile response time metrics can be predicted with a proper level of precision, given a mean response time prediction.

Bi et al. (2004) analysed the advantage of wavelet packet and proposed a novel load forecasting algorithm based on wavelet packet analysis. Based on the insight of the wavelet theory, the bi-orthogonal wavelet and symmetrical border extension method are selected to improve forecasting performance. The algorithm developed provides information for analysing characteristics of load components, and the forecasting accuracy is improved.

3.10 Quality of Service

Menascé & Almeida (2002) argue that in a web environment a user does not care about traffic jams, overloaded servers, network bandwidth, or other indicator of system activities. Besides content and aesthetics, online users want performance, availability, and security. To an online customer, quality of service means fast, predictable user-perceived response time and 24 x 7 uptime. And degradation in the service level of a website is noted in real time. Quality of Service (QoS) indicators for web services should represent response time, availability, reliability, predictability, and cost.

A study by G. Ferrari et al. (2006) classifies several existing studies on the Quality of Service of e-Business systems into three categories. The first category treats the

system as a 'black box'. The response of such a box is measured and a policy that controls some operational characteristics reacts to observed changes on demand. However, studies in this approach do not attempt to predict the performance of a system in relation to its structure. The second category relies on *a series of experiments* whereby emulated clients are used to calibrate the behaviour of the system. The statistics gathered during the experimental phase are used to configure the system for the required performance metrics during run time. Although there is an element of prediction here, it is limited by the fact that only situations similar to those encountered during the experiments can be effectively managed on-line. The third category includes *studies based on the analysis and evaluation of models* whereby the system is represented by some kind of a queuing network.

The study by Bouch et al. (2000) was designed to investigate users' requirements for internet Quality of Services. The study lists a set of objective thresholds that reveal users' subjective assessments of quality. It shows that:

- The task to which users are committed, the length of time they have been interacting with a site, and the method of page loading that affects the acceptability of QoS.
- Tolerance of delay is affected by users' conceptual models of how the system works.
- Poor website performance leads to low-grade company image and often compromises users' conceptions of the security of the site.

There is a great number of participants in the design of internet services: server designers, network providers, advertisers, content providers, and the end-users themselves. A failure to understand users' on-line QoS requirements may affect users' conception of a company's stature and commercial viability which, in turn, affects the business interests of service providers and advertisers. The future internet will have more users, and support a greater variety of internet applications. It has the potential to change the way in which consumers interact with companies.

3.11 Conclusion

The access log analysis does not necessarily produce valid results of usage due to the rapid growth rate. The maintainer faces the problem of improving the calculations by intuitive conjecture and educated guesses. Even then, the long period estimations of usage are rather lucky guesses instead of fact-based knowledge. Still, the development process has to be done several months or even years pro-actively by the terms of hardware and application investments.

The general interests in performance management are to create an automated process which reduces manpower in routine work and proactive operations. Manpower is still required in a setup of the management process and the goals of the operations. The proactive activities are required especially in the environment of e-commerce in order to minimise damages. In general, competition for market shares is strict and in case of dissatisfaction by customers, a switch to the competitor's service is oversimplified.

Performance analysis is mathematically considered to be simple. The basis for the analysis of a website in service is rather complicated. The throughput on some incoming rates of requests should be known beforehand. However, the throughput is not constant due to several reasons. Hence, the performance analyses have to be updated regularly to match the present requirements.

Another essential measure of performance is the response time. For the user, it is the first indicator of incipient problems. At first, the user feels no specific slowness in the responses and thereafter problems heave into view. The most observable indicator is an error message. However, there are two unsolved problems in the measurements; first, the end-to-end response time. It varies depending on transmission route and even the time of the day. The second problem is the rendering time in a browser. The competition between browser manufacturers intimidates developing of continuous faster solution to ensure market sharing. At the same time, the content producers create still more attractive contents to seize on the improvements of browser capabilities.

Continuous monitoring of performance is necessary due to the changes in the internal state of the system, the updates of the applications, and the changes in

usage. Therefore, the selection of the monitoring software is troublesome. Several theoretical models are shown, but the monitoring tools in practice are mostly based on the monitoring of active state. The monitoring, by regular requests of simple and static URL's, does not produce accurate enough data for proactive operations of website maintainers. However, operating systems are embodying tools for monitoring consumption of resources.

In general, the performance of the system has to be good enough during the peak time to satisfy user needs. To figure out the moment of the hotspot and top usage, the log files have to be analysed. In this section, the analysis methods and the sample collection processes are discussed. In addition, the prediction model of usage in the near future is shown at the end of the chapter.

4 Access Log Analysis

This section describes the log file information based on the actual usage of the analysis. The log data will provide a clear understanding as to when the service was used, how big was the instantaneous peak load, and the kind of service the site has been used for.

Queries made by users are recorded for all the application servers or all the load balancers. Necessary information for the user's requests is stored and analyzed in order to find the service most frequently used applications and the required parameter. Applications and the relevant information presented in the log information, in which users are using the service, are included in the load tests.

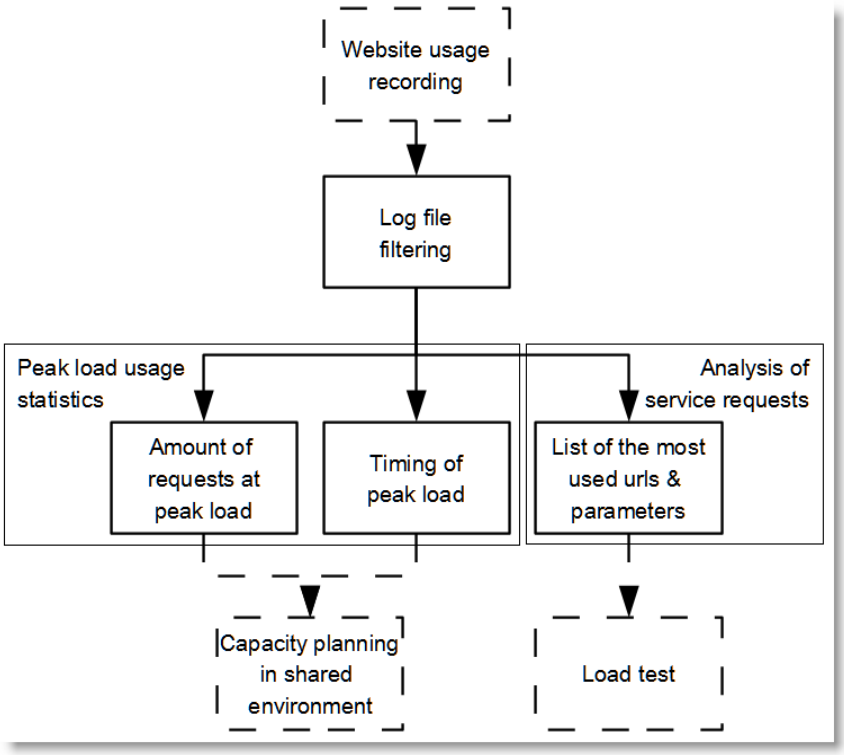

Figure 4.1: Rescisco-model for log file analysis on websites

The log analysis model used in this study is based on a study (Soininen & Jaakkola, 2012) and is presented in Figure 4.1. In section 4.1, the web servers' own log file recording process, and the mechanism for data pre-processing are discussed. Then, section 4.2 presents the height of the peak load with the timing and number of service requests and their parameters. Section 4.3 introduces the analysis of query

types. Section 4.4 discusses the trend analysis of daily peak load. In section 4.5, the sensitivity of the analysis is briefly discussed, and section 4.6 finally draws this chapter to a close with the conclusion.

4.1 Collecting and Sampling Process

E-business workload is defined in Menascé et al. (2003) to be made up of sessions. A session is a sequence of requests of various types made by a single customer during a single visit to a site or a system. During a session, a customer requests the execution of various functions such as browse, search, select, add to the shopping cart, register, and pay. A request to execute a function may generate many other requests to the system. For example, several images may have to be retrieved to display the page that contains the results of the execution of a function.

The log files are collected during the normal usage of the web services. In this study, the web services are based on Apache (Apache Software Foundation, n.d.) or Microsoft's Internet Information Service (IIS) (Microsoft Corporation, n.d.). The impact of the log file collection process itself on the performance of the system is not adequately discussed in studies. Empirically, it is known that in some Microsoft operating systems, the huge log files are deteriorating the whole system performance due to the writing to the log files. Only a few remarks can be found (DeveloperSide.NET, n.d.; Microsoft Corporation, n.d.) which consider the log file size. Therefore, the log file rotation, i.e. creating new files regularly, has to be considered to avoid decline in performance due to the measurement process. At the same time, we assume that the log file writing process does not load the system remarkably or merely affects only one layer on the system. Probably, the layer mostly affected is the load balancer group, which is only lightly loaded in quite many cases.

As to the result of the analysis, we are only interested in the peak load moment and the number of requests during the hotspots. In general, the peak load moment happens in the same period, e.g. daily basis. On the other hand, the duration of the peak load time varies, even from day to day. It depends probably on quite a

number of things, e.g. the content, response time, day of the week, season or even weather. Within the long period, the peak time varies and even new peaks can be found. Despite the changes in the peak load, the number of the arrival requests is the key issue. All the requests have to be satisfied within the required response time, and supplied content to the request has to be relevant and valid.

The size of the access log file is easily very large to analyse as such. Each user session generates even hundreds of requests to the server system. E.g. in the case of one million unique browsers per week, there could easily be more than ten million requests per day. It is not inevitable to analyse all that data; instead, five to ten minutes hourly per day is enough. This is based on two assumptions. Namely, sufficient data is available per session and the load is even distributed to the whole one hour period.

The log file has to be filtered out from the samples generated by the preceding natural loading process. This ensures that the loading process itself does not infect the result of the usage analysis. In addition, all requirements that are not fetched by the system under study have to be filtered out such as requests relied by the load balancer even though generated outside of the system under study. And therefore, they do not generate real load to the system.

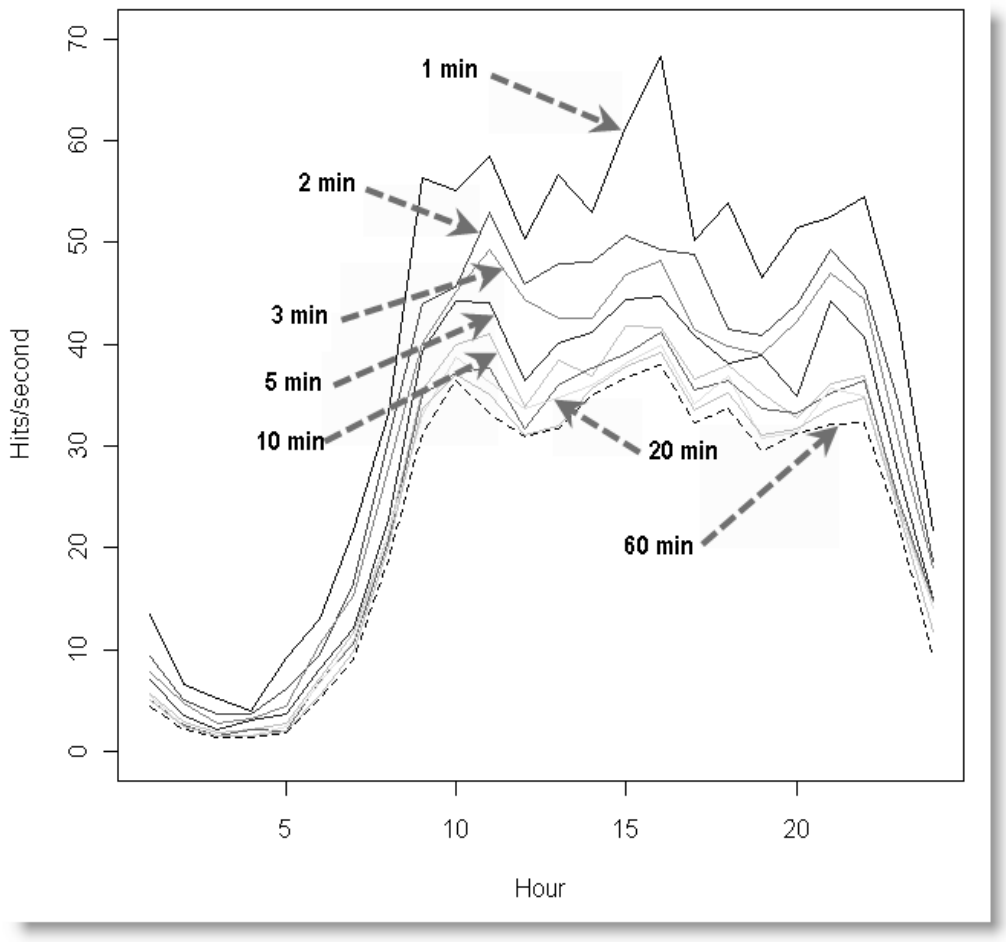

Figure 4.2: The effect of sample length

The effect of the sample length is shown in Figure 4.2. As we can see, using 1 minute sample per hour, the average maximum usage in hits per second is 68 at hour 16. By increasing the sample size to 60 minutes, the average maximum usage is decreased to 38 hits per second at the same position. However, by shortening the length of the sample, the possibility to the outliers is increasing. We can estimate that a 5-minute sample size eliminates temporary outliers and is simultaneously short enough for performance estimations.

The number of arrival requests in time in Figure 4.2 is defined as follows:

$$\lambda = max(n_m)_t \tag{15}$$

, where n_m = number of transaction per second at time slice m, t = hour, 0...23.

4.2 Arrival Rate

One aim of the access log analysis is to figure out the characteristics of the peak time using the most effective means. It can be done using some statistical methods. Often, a cyclical pattern in an access log can be found. In different days, the peak is normally concentrated on the same time of the day. In many cases, the load during a weekend is different from during workdays. The difference depends on the nature of the service. On media services, the usage is lower during the weekends and for entertainment services the usage is higher in the weekends than during office hours. The analysis can be done on a daily, weekly or monthly basis. The suitability of the following analysis methods is performed: decomposing, linear regression, and ARIMA methods.

Box & Jenkins (1976) suggests that this kind of data is usually best achieved by a three-stage iterative procedure based on identification, estimation, and diagnostic checking. *Identification* means the use of the data, and of any information on how the series was generated, to suggest a subclass of parsimonious models worthy to be entertained. *Estimation* means the efficient use of the data to make interference about parameters conditional on the adequacy of the entertained model. *Diagnostic* checking means checking the fitted model in its relation to the data with the intention of revealing model inadequacies in order to achieve model improvement. In this context, identification refers to the sampling of raw data, and estimation refers to the analysis of the sampled data. Finally, the diagnostic checking refers to the checking of the validity of the developed model.

The number of requests based on an access log is shown in Figure 4.3. The time scale is 96 days, 24 samples each, totalling n=669 samples. However, there are some outliers, i.e. in the level of 50 up to 70 requests per second, which should not be in an essential role while dimensioning the system. In the figure, the number of requests per second is a mean value for one hour, and it is based on five-minute samples.

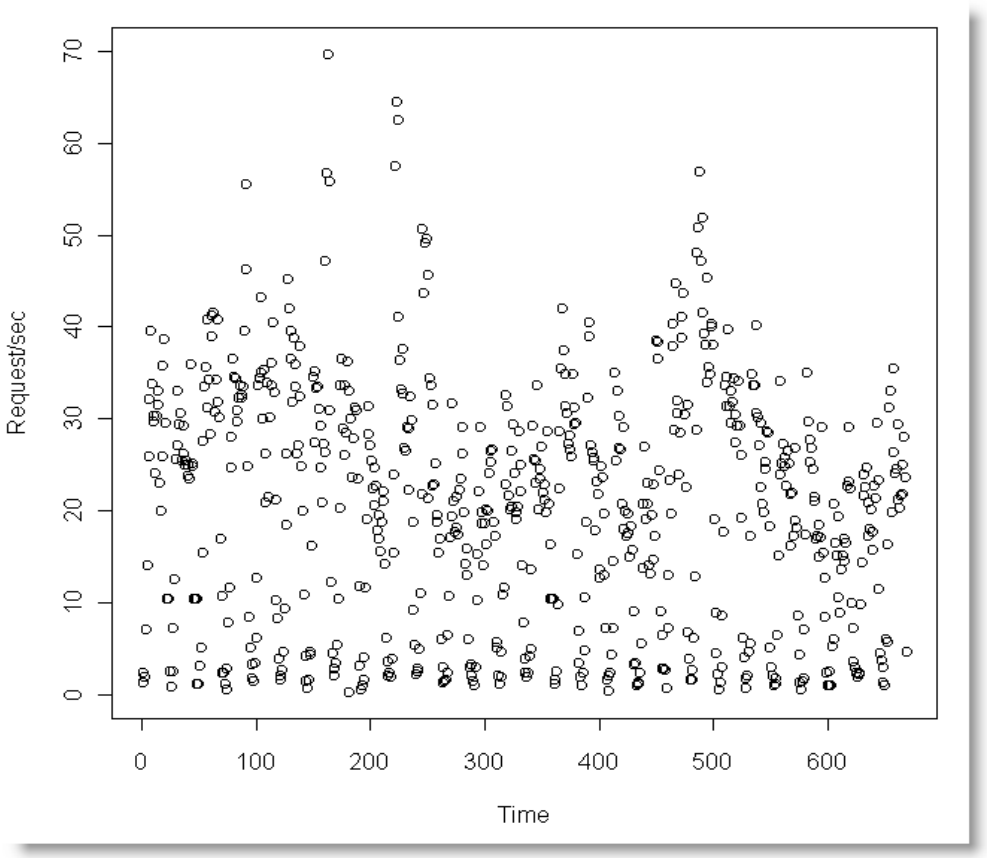

Figure 4.3: The hourly maximum arrival requests per second on a period of 96 days on an e-commercial website.

For website modelling, it is essential to know the service load on the peak size and its timing. If there is more than one service using the same server system, the log

Table 2: Sample of peak load height and timing

Date	Time	**Requests / hour**
20120310	00:00:00	121056
20120310	01:00:00	227064
20120310	02:00:00	332808
20120310	03:00:00	307200
...	...	...
20120310	21:00:00	74136
20120310	22:00:00	51720
20120310	23:00:00	82464

analysis has to be performed on all services in order to obtain sufficient information. As a result, the level of loading for each period is summarized in Table 2. The result is easy to communicate and is readily available for further processing. In consecutive days, it will form the time series, allowing for the drawing up of a forecast for the future use of the service.

As an example, Figure 4.4 shows the typical workload during a single day. The figure consists of 24 mean values of samples for each hour. The most relevant workload occurred between hours 7 and 19. The web service performance has to be fitted according to the maximum of peak time. The mean problem of sampling of arrival requests is extracting outliers out of the relevant data.

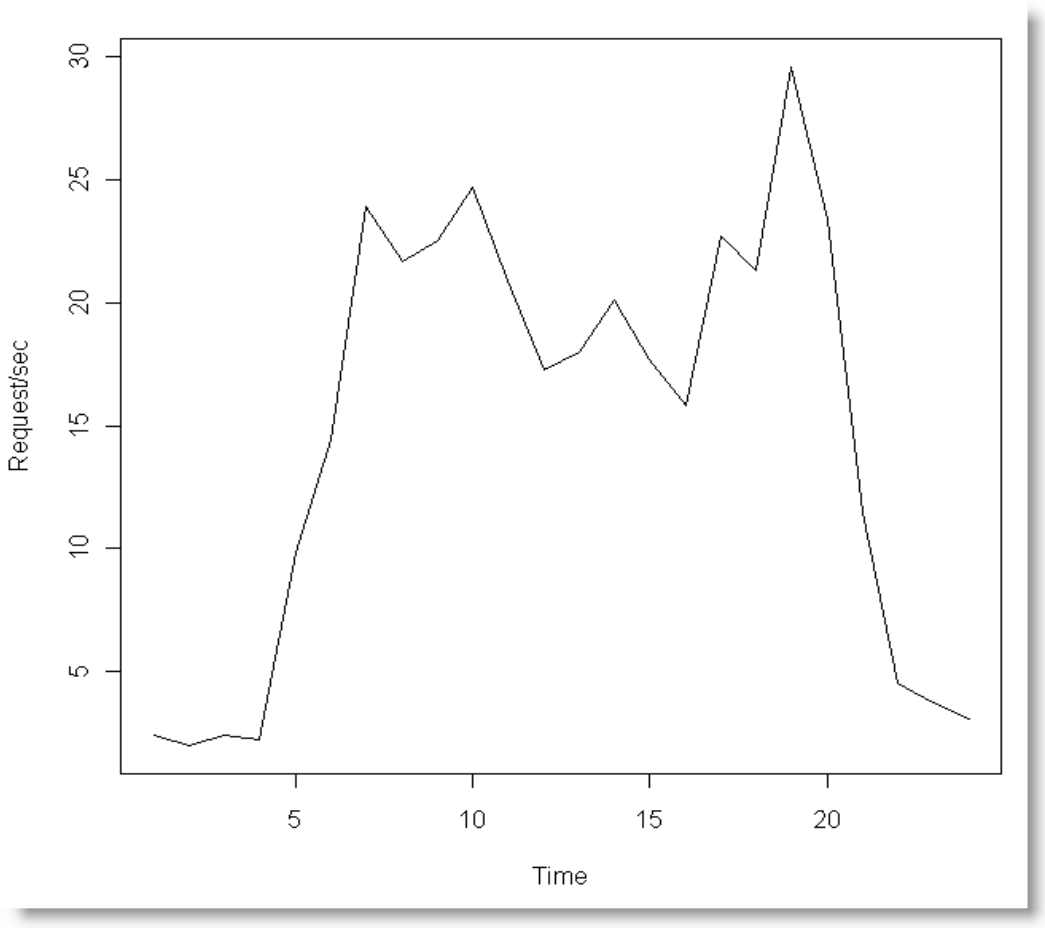

Figure 4.4:The maximum workload for each hour during a typical day

The general purpose of the different smoothing methods is to remove outliers and anomalies from the data and extract the most meaningful values to fit the optimal system. The first method is decomposing. The *stl* is the standard function in the statistical computing application, R (Team, 2011). The method is a two-step procedure in the R. First of all, the raw data vector has to be converted to the time series. The only remarkable parameter is the number of observations per time unit. In this case, the time unit length can be chosen between 24 hours and one week, i.e. 168 hours. On the second phase, the decomposition of the time series requires two span parameters, one for trend component and one for seasonal component. Those parameters have to be synchronized to the time unit length. In this matter, 168 hours is used. The result is seasonal decomposition of the time series.

The second method for analysing the access log is decomposing by *Loess*-method (R. B. Cleveland et al., 1990). The method produces similar seasonal decomposition as the *stl*-method. Let λ_v be a vector of existing values for arrival rates and the $v=1...N$, where N=number of observations. The *loess*-method performs decomposition to the trend component, the seasonal component, and the remainder component, denoted by T_v, S_v and R_v:

$$\lambda_v = T_v + S_v + R_v \tag{16}$$

The additive model is more suitable in the case of constant seasonal amplitude. Another possibility is to use the multiplicative model as follows:

$$\lambda_v = T_v * S_v * R_v \tag{17}$$

The result of decomposing is shown in Figure 4.5. The total length of the data is 90 days with 24 samples each day. The raw data is shown in the most upper part of the figure, in the data panel. In the seasonal panel, the decomposed seasonal data and the trend component are shown respectively. The remainder component expresses the anomalies and outliers.

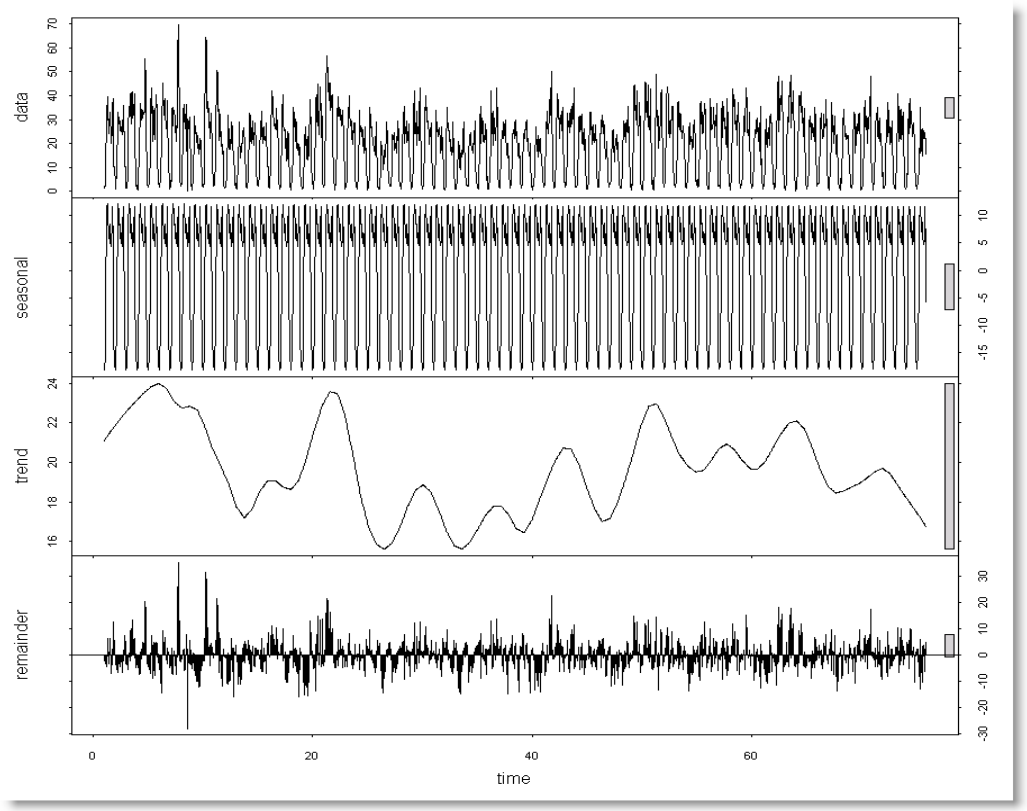

Figure 4.5:The decomposed actual usage of web service

The daily maximum values are the most dominant values for system measurement, and they are expressed as:

$$Z_i = max(S_i, S_{i+1}, \ldots, S_{i+23}) + max(T_i, T_{i+1}, \ldots, T_{i+23}) \tag{18}$$

where $i = 1 \ldots \frac{v}{24}$. The daily maximum can occur at any time of the day. This leads to time series, which is not equal in intervals. By changing the time series to daily maximum values, the time series can be equalized without any risks.

The result of decomposing is the sum of these components, and it has to be changed to prediction. There are several methods to predict further. However, in this case we will use two of them. First of all, the extrapolation based on the linear regression. Secondly, the sum curve is extrapolated in the form of a curve. The results are shown in Figure 4.6.

In the seasonal decomposition, the Z_v is the sum of the trend and the seasonal curves:

$$Z_v = T_v + S_v \tag{19}$$

The prediction for the Z_v is based on the Fourier series approach:

$$y_t = a + \sum_{k=1}^{K} [\alpha \sin(\frac{2\pi kt}{m}) + \beta \cos(\frac{2\pi kt}{m})] + N_t \tag{20}$$

where N_t is the ARIMA process. The value K can be chosen by minimizing Akaike's Information Criterion (AIC).

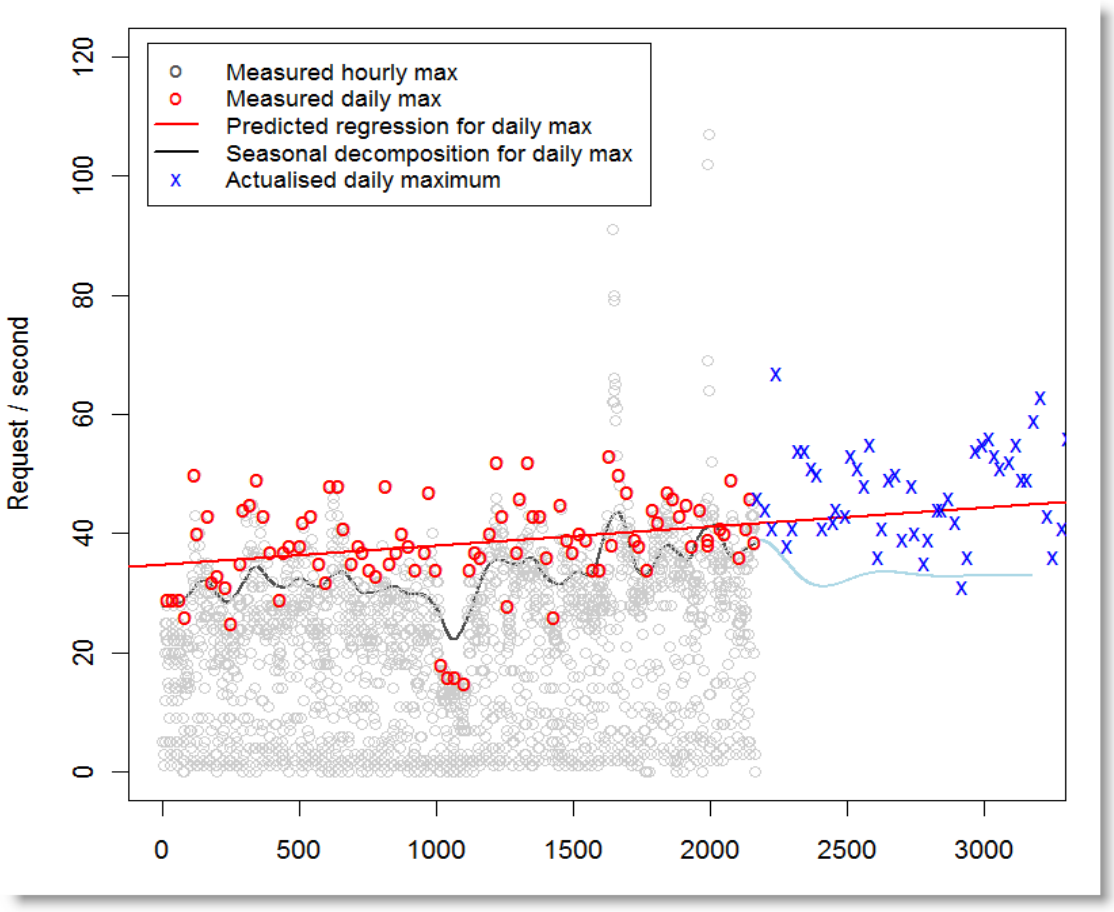

Figure 4.6: Prediction for daily peak values using seasonal decomposition method or simple linear regression based on daily maximum values

In the case of linear regression, just one maximum value for a day is generated to characterise the usage for the linear regression. As we can see in Figure 4.6, the decomposition curve is continuously on a lower level than linear regression. The

decomposition method aims to leave out all the high values. However, in this case the high values are the key issue, and they cannot be removed until they are outliers.

The decomposing method is somewhat arbitrary. The different components produced, are not real, nor are they measurable by any means. At the same time, the results cannot be validated by statistical means. In addition, by changing the time series and decomposing parameters, the results vary remarkably.

To summarize the above, the most suitable prediction method for evaluating the maximum values is the simple linear regression. That method can be used to find the maximum values, and simultaneously and effectively remove the outliers without distorting the results.

4.3 Analysis by Types of Queries

The access logs have to be analysed also by qualitative means. The total number of queries in a certain time frame is playing an important role in performance evaluation. However, most of the queries are unique, and get different response times from the service system. There are two types of differences, the applications are either different or they have different parameters.

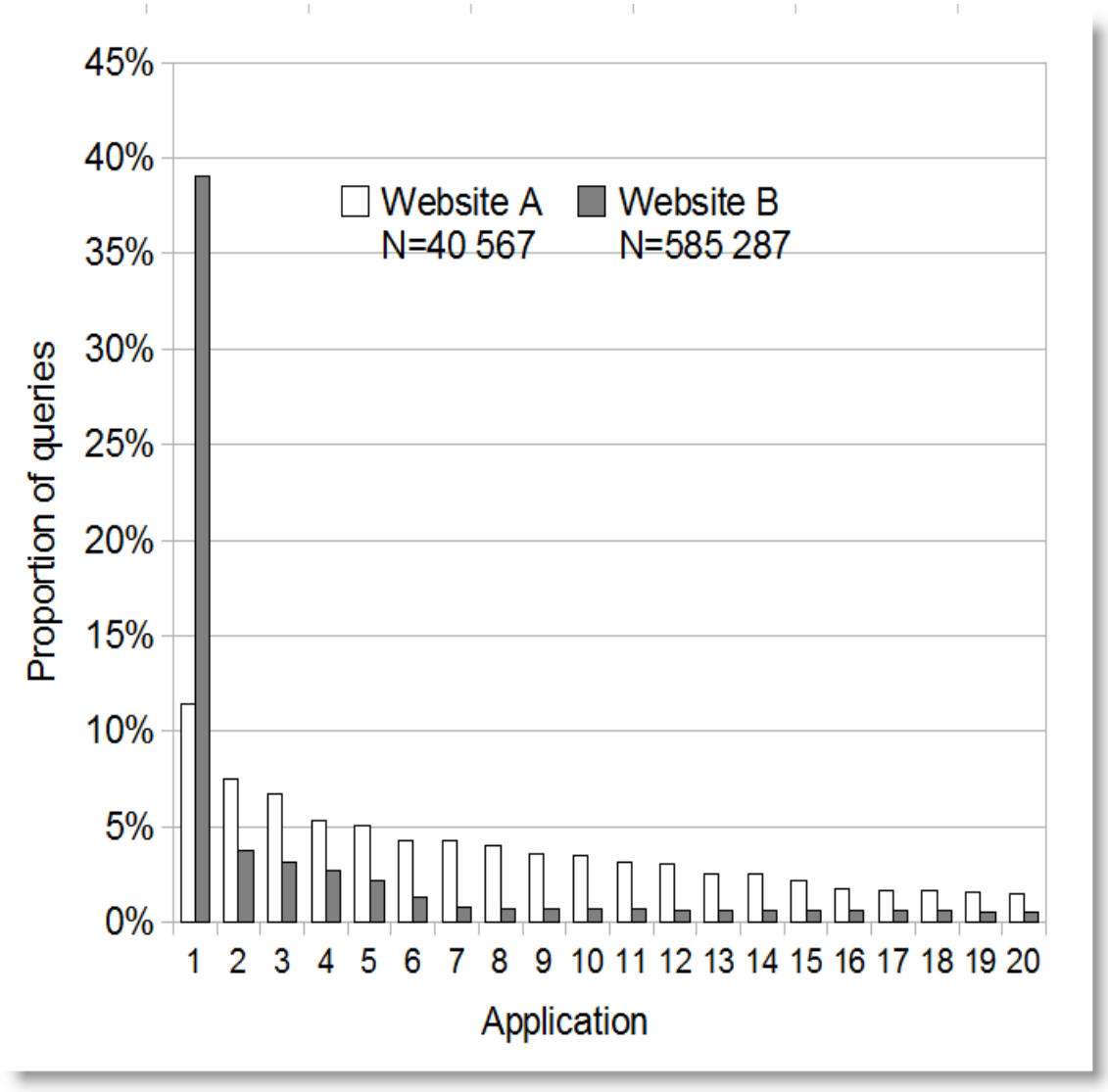

Figure 4.7: Distribution of application queries at two websites on a typical day

We have analysed the access logs into two, as service e-commerce websites *A* and *B*. Both websites use different applications. In Figure 4.7 and Table 3, the numbers

do not refer to the specific application but the popularity of the application on the site. Figure 4.7 indicates the proportion of the applications. The first 20 applications are shown, and the rest creates a heavy tail.

Table 3: Number of parameters on application at the websites A and B

TOP-20 Application	*Website A*		*Website B*	
	Parameters	*Number of Combinations*	*Parameters*	*Number of Combinations*
1	0	-	3	huge
2	19	huge	2	huge
3	10	huge	1	huge
4	0	-	0	-
5	5	huge	0	-
6	10	huge	0	-
7	10	huge	1	huge
8	0	-	2	huge
9	0	-	0	-
10	1 / 11	2 / huge	0	-
11	5	huge	2	0
12	17	huge	0	-
13	0	-	0	-
14	0	-	0	-
15	22	huge	0	-
16	2 / 3	3 / 3	0	-
17	14	huge	0	-
18	0	-	0	-
19	0	-	0	-
20	0	-	0	-

The number of parameters and their combinations are shown in Table 3. In most cases, the number of possible values of parameters is not known, hence the total number of combination is not known. Therefore, the "*huge*" in the table means a great number of different parameter combinations, from several hundreds to thousands. The number of combinations depends on the number of parameters and their number of allowed values.

Applying the most used 100 applications with the suitable parameters to the source of the natural load, the realistic load can be generated (Soininen & Jaakkola, 2012). Using a large enough number of queries and set of parameters, the effect of cache can be eliminated in the server system. Otherwise the analysis can be distorted due to the short response time and resource consumption when the minimal number of pre-set parameters and low number of applications queries are used. The process of natural load is described in the Chapter 5.

Table 4: Sample of most used URLs with parameters

/ads/?departmentId=5
/WidgetSettings.aspx
/news/Default.aspx?newsid=299221
/js/AdCounter.js
/news/glow/1.7.7/widgets/images/darkpanel/bg.gif
/news/Default.aspx?newsid=266704
/news/DynamicImageResizeHandler.ashx?image=442d576b-b0ed.jpg&width=320&height=320

As to the second result of log analysis, Table 4 shows the result of the qualitative analysis of the resulting URL and related parameters. The actual protocol and web address may be omitted if they do not cause confusion. When the service requests are placed in a random order, it can easily be used for the source file in the load tests. When the service requests and their parameters in Table 4 are used for load testing, it confirms that the website under test is loaded with exactly the identical proportion as real users do. In addition, service requests and their parameters are real, occurring in actual usage, and are not programmatically generated random combinations.

4.4 Trend Prediction

The trend on prediction of the system usage is a vital component for estimating the behaviour. There are mainly three different methods for prediction: a) tendency or dependency is constant ($y_t = b_0 + b_1 x_1 + b_2 x_2 + \dots b_k x_k$ and $y_{t0l} = b_i$ remains as constant); b) leading indicator; or c) structural analysis. It is very difficult to find a phenomenon as a leading indicator which has a good enough correlation to the single website. On the other hand, the general structure analysis is required to be analysed for all websites due to the different applications. It is not possible to perform for all applications and all versions. Due to these reasons, the prediction is based on the continuance of tendency in the model of this study.

In this study, we are interested as much as possible in generating a forecast for the future use of the service. Usage of the service prediction is an advantage if it is the most conservative. In this way, sudden seasonal changes receive the lowest weight. We cannot take advantage of more than 90 data points to describe the use of the service history, since sometimes the website usage is changing too fast and in such a case, the forecast reliability decreases sharply. In addition, we cannot predict a longer period than what is the number of data points in history. On the other hand, the prognosis needs to extend far enough that the operational management and maintenance steps can be planned and carried out during the forecast period. When using the 90 data points backwards, it can be predicted from about 30 data points on. As shown in Figure 4.6, a method which best describes the situation is a simple linear regression. The forecast accuracy is not increased even if the seasonal decomposition method is used to filter the daily maximum values.

4.5 Sensitivity Analysis

The log data analysis is related to the outcome of the accuracy of a few factors. This section seeks to identify the most crucial ones, to evaluate their impact on the outcome, and to assess the likelihood of their occurrence, if relevant.

First of all, the web application structure is meant to use the application to call an URL or its attributes. The service can be constructed in such a way that the data is placed in a file system, in which case the invitation is a direct reference to the file on the disk. The service can also be built so that the URL is a call to the desired application, and the file is defined as an attribute. Therefore, we have taken into account the log analysis in both formats. In this study, the result of both methods is taken into account.

Secondly, the use of the service associated with seasonal fluctuation is greater than that generally used in the calculation period (for three months). This may cause significant changes in the forecast changes. We need, however, over many years collected data using the service before we can accurately determine what is the load peak associated with seasonal variations and what is the individual random variation. In this study, the data of website usage were not available for a sufficiently long to permit a seasonal variation in the time series could be taken into account.

Thirdly, the amount of data being collected affects the accuracy of the sample, and which describes the use of the service. The key issue is, how many unique URL requests are picked according to the sample which describes the entire use of the service. And also, how many key types of URLs are selected. In practice, it has been observed that the general use of the service is on a long tail. Therefore, a significant proportion of the requests directed to a few pages or applications, while most of the pages or applications are directed only a few requests. The same also applies to the URL attributes. The study has been used in the default value of 100 types and 2000 unique URL attributes. Variation in these values does not provide a substantial variation in the outcome.

4.6 Conclusion

The usage of an e-commerce web service can be analysed based on access log files. The key issues in the analysis are hotspot usage and the most used queries with real parameters. The resolution of peak usage has to be balanced between short-

and long-term accuracy. Using very short-term, the high peaks can be revealed, but the result can be sensitive to disturbance. Instead of using very long terms, the mean value of access can be revealed, but the result can be very conservative against changes. The result of analysis is utilised on a simulated natural load which is discussed in the next chapter.

In this chapter, we have presented a method that can be used to form a true and fair view of network use. The result is a straightforward pattern of log file which is easy to communicate. It shows the daily load peak height and its timing. In addition, the analysis can be used to describe the type of queries generated by users.

5 Mastered Way of Workload and its Impact

This chapter presents the workload and resource consumption measurement methods used in this study. The system under test (SUT) is introduced as a black box, whose internal details are not known. Every single server resources are monitored in each load case, and they can be visualized in an appropriate way. Monitoring of resources is aimed not to increase the load of the resource or to occur as little as possible.

Thalheim & Tropmann (2011) suggest that the service applications, the history of the usage allows us to conclude that the usage typically occurs in a particular repetition, and are used during certain periods. They may be characterized by the consumption of resources and behavioral measures. Therefore, we can assume that at certain times, the future of these processes are running in the same way as they have been in the past. It can consequently be expected that future behavior can be described by conduct made in the past.

The practical arrangements for testing have been carried out in several different independent commercial online services. Services are being loaded, and the resource consumption is controlled as described in this section. All services have been in commercial use throughout the duration of testing. Architecture is shown in Figure 5.1(A) and Figure 5.1(B). Figure 5.1(A) shows a typical commercial application architecture of the multi-layer model. The system is easily scalable by adding complete individual servers or components to increase efficiency. Figure 5.1(B) type is a simple small-line server system typically used. It does not include the redundancy so that loadbalancer layer is not needed. It can scale worse than the A-type service, and increased performance is obtained by changing only the complete servers more efficiently.

The A-type service can be categorized as the *computing cluster*, wherein each node produces separate and autonomic functions and does not transfer data between parallel nodes, which takes place only in different layers. The load balancing and application server layers in the system can be classified as the *high-availability cluster* due to the redundant nodes. In the case of failure, a single node of the

server does not prevent the operation of the entire system, only performance is reduced by the amount of the failed node.

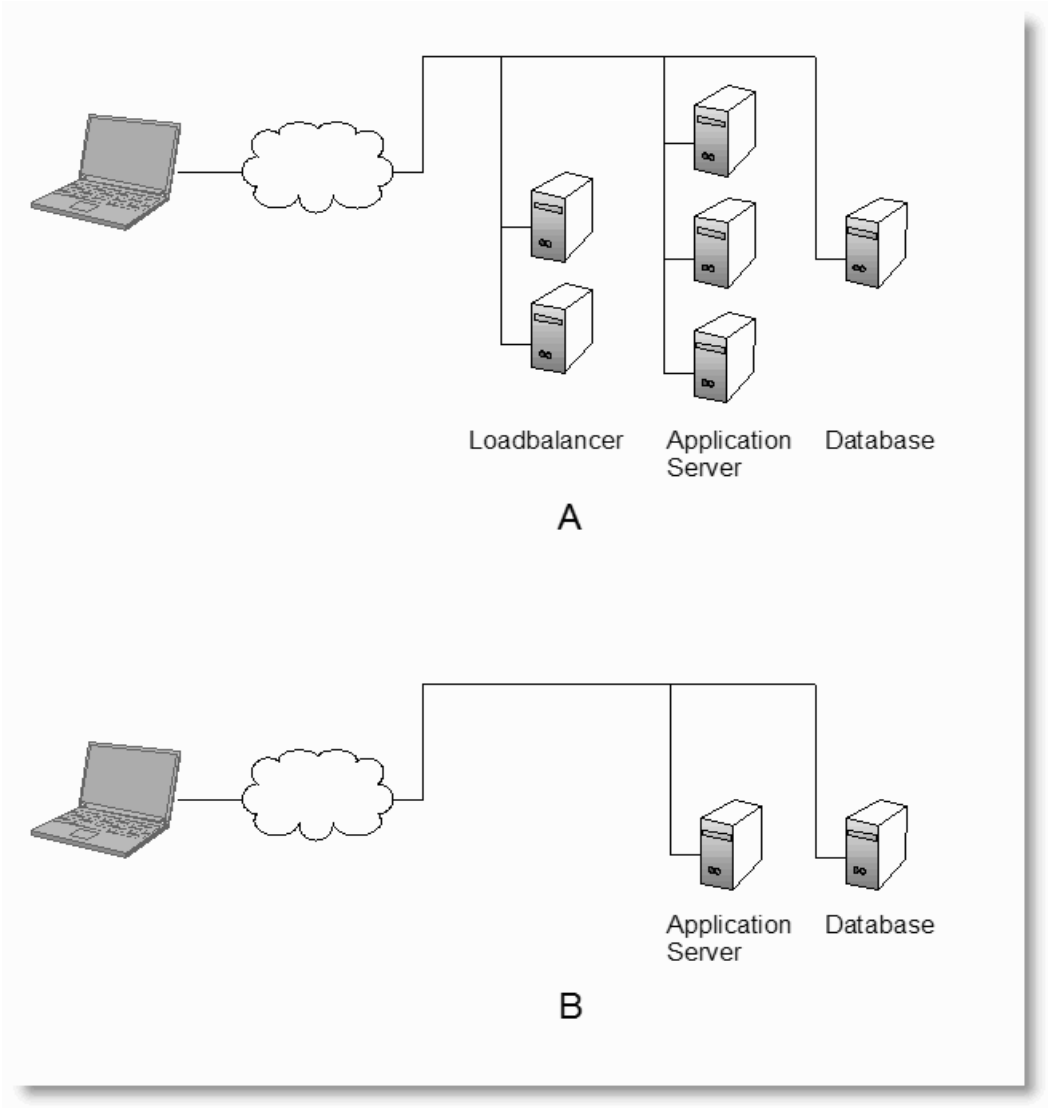

Figure 5.1: The architecture of two separate test environments used in this study

Determination of system performance occurs in several stages:

- Loading the system with natural queries. Details of the testing activities are discussed in Sections 5.1 and 5.2.
- Load-time resource usage monitoring: the result is obtained for each server resource utilization and workload, presented in more detail in Section 5.3.

- Correlation between workload and resource consumption: the result is maximum system performance. The descriptive measure is defined in Section 6.1.
- The current spare performance at the time of analysis is a result of a review data consolidation. The matter is dealt with in Section 6.8.

Knowledge of the current performance compared to the expected regular top-load is essential to optimize server performance of the system's point of view. By combining the usage analysis of the actual system performance, it gives an adequate understanding of the current excess performance.

5.1 Impact of Test Load

Loading the SUT by the well-known load is being reviewed as well performance of the system of critical resources and the correlation between the workload and resource usage. Consumption of resources caused by the workload is discussed in Section 5.2. Another factor which exists during the test is the undetermined background load, which is caused by users doing the actual load of queries during the test. The significance of the backround load can be minimized (but not completely removed) by timing the test at the moment when the background load is minimized. In many of the practical online services, the lowest possible background load occurs at night. Uncontrolled load is another potential source of internal system maintenance operations, such as backup or database management tasks. Usually, these service operations should be done on a well-known schedule and are also well-known for their duration. This background load which is caused by the service runs can be avoided by timing load testing appropriately.

The correlation between the load and resource consumption is also affected by the factors that can be detected, but their magnitude cannot be affected. In the long-term, software aging can be detected, but it cannot be influenced by testing arrangements. The only possibility to minimize the impact is a server or an application, to restart or even a full system restart. In general, applications should be avoided when restarting to keep interference with the system as little as

possible. On the other hand, application can be utilized and the internal caches can be used as efficiently as possible. The effects of software aging phenomenon will be discussed in Section 6.7.

Any control requires that the load factors can be controlled during the test load, such as queries, variation over time, the timing of peak load and the amount of it, different number of inquiries and the amount of data transferred. This is discussed later in Section 5.1. The second group consists of random factors. Partially, they can also be interpreted as being hidden factors. They affect the activity, but they are not known accurately, or the value cannot be measured. This group consists of a variety of factors, including internal state of the system, the system resulting from the history of usage, the need for communication between different nodes of machines, and the complexity of questionnaires used in load testing, even their order.

The test load is presented as having both a quantitative and qualitative impact on system resource consumption. The quantitative impact means the number of queries over time. The qualitative impact means the differences in queries resulting in load variation. Synthetic load may be used to determine system performance, but the queries do not match the actual queries made by users, so the resource consumption is not responsible for the actual use of the situation. The better descriptive situation of the reality created when a synthetic load is used instead of authentic is to log the recorded real queries made by users.

A too-low test rate with a load testing, actual use of resources, and the correlation between consumption may not occur correctly. Resource consumption may change in different ways at different load levels. The operating system kernel has been used for the acceleration of the system characteristics by optimizing the use of resources, such as the available memory using the I/O cache. In this case, the memory consumption at low load is not at all correlated with the actual load. When the load increases, it is found that the consumption of memory appears to remain constant, and is then released from the cache memory to serve for use in increasing the load.

The impact of quality means that the formations of response are used for different applications by different queries. Each application uses its own resources in a typical way; URI can create a complex database query and a database server to consume processing power and result in disk I/O, but does not, for example, consume any file server resources. The change of parameters in queries can cause significant changes in resource consumption. Typically, changes in the parameters influence the results of search queries, and they will be implemented through the database. Database indexing and implementation of query application will depend on how the change will affect the parameters of the database server resource consumption.

Testing the performance requires several test runs of long period of time. Contents are updated on a regular basis for many services, hence URI is not identical with contents of the various tests or the same content in different URI. In this case, the different responses obtained from the tests are not the same and cannot be directly compared. From the user's point of view, however, the service availability in different load conditions is a key issue. Therefore, the accessibility of the service has to be compared and not with the identity of the response. Assessing the response of the service accessibility instead of the time in different test runs is comparable to the results from the response contents.

User interest in the change of the service sub-region to another, in some cases a rather short period of time, can cause a change in load between the two applications. This in turn leads to the use of the characteristics and the system will be further changed in the consumption of resources. Log analysis can confirm that the server system is loaded in the same proportion as the user load, and thus will lead to the correct understanding of the server resource consumption.

As the test load increases, response time will also change. The hypothesis is that response time does not necessarily increase linearly with increasing load. At the beginning of load increases, it may increase slowly until at some point it begins to rise sharply, as shown in Figure 3.5. Assuming further that the effect of the test load is achieved, a linear response is only in the event that the resources need to queue waiting time, and increases linearly with the length of the queue or amount

of use. This situation may be, for example, that the use of the CPU and memory usage does not slow down the increase in service activities.

5.2 Controlled Load on the System

When the system is loaded with the highest expected performance of the system, there are at least two problems. First, in the maximum load or close to it, the system is not stable. It appears as random or cyclical high spikes within a response time. This is because when running out of resources, they have to queue or limit the overloading of the service definitions to take effect, then the user will have to queue even for access to the service. Instead of the usual contents for the user, visible messages are notifications of the state of the system, not the user's desired content. On the other hand, too high a degree of loading can cause disorder mode as is also shown in the service user, either in response time which is too long, or even receiving a server response to be other than the content desired by the user. Service, which is in commercial use, interferes with the operation of the testing, and is not acceptable. Even a momentary overload can cause interruption on the payment transaction and could result in an awkward manual processing and the dissatisfaction of users. Hence, regular testing is needed of the normal operation of disturbing causes, and possibly, the continuous dissatisfaction among users.

Later, it is assumed that the load can be carried out, amounting to the service on a regular basis by loading in a known amount of virtual users. In such a case, the requests are known, as well as their contents and the time when the request is sent to the server and response time. A common use of the service is, at the same time, possible to test with, but it should be kept to a minimum so that the unknown magnitude of the load can be minimized.

Figure 5.2 shows the connection between the number of virtual users and the number of incoming requests to the server. HTTP is a connectionless protocol, hence the number of concurrent users is not a well-defined concept. Even when using the HTTP protocol version 1.1, and the Keep Alive feature, the number of concurrent users is not unique. As can be seen in Figure 5.2, the number of users

(0 ... n) can be any at time *t*. In the future, *request per second (rps)* is used to replace the number of virtual users, since it is unique in any given time.

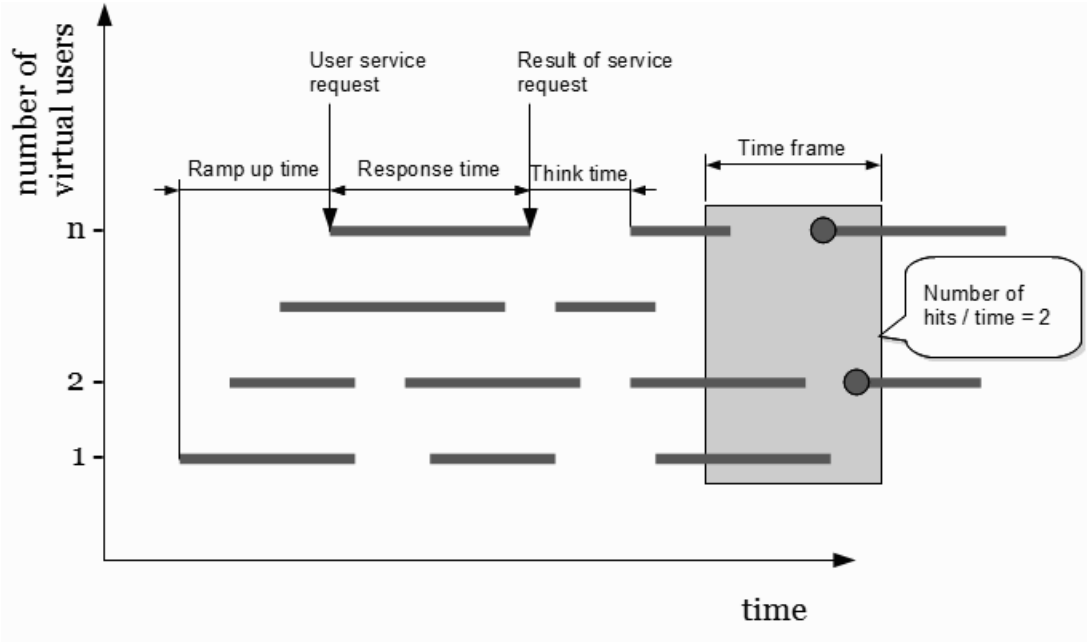

Figure 5.2: Schematic representation of the number of requests / time and number of virtual users

The idea of gradually changing load rate during loading is raised at first in Andreolini & Casolari (2006). In the model shown in this study, it is modified so that it is better suited for use in automatic test load.

Figure 5.3 shows the structural load used in this study. The desired load level is determined indirectly, based on the number of virtual users. They do not match exactly. The virtual number of users is only about how many requests the server can cope with. In practice, however, response time and thinking time cause that part of the virtual users to wait for the response from the server, before thinking time has expired. So, the active users of the service is less than or equal to the maximum number of virtual users.

Figure 5.3 shows a single loading test with an overall length of about 1300 seconds. For each pulse duration is about 180 seconds, and after a pause between the pulses, it is approximately 40 seconds. During the break, requests are not generated by means of the service load at all. Figure 5.3 shows that the load is increased progressively to about 45 rps level until it reaches an empirically determined service-specific maximum level, which is here about 210 rps. The load

may also be degressive. Maximum load is chosen so as to cause no more than about 40% to 50% utilization of the resource, which is the bottleneck when a normal usage is still possible. The load test is performed once a day. The system is operated and maintained in accordance with normal procedures between the two tests.

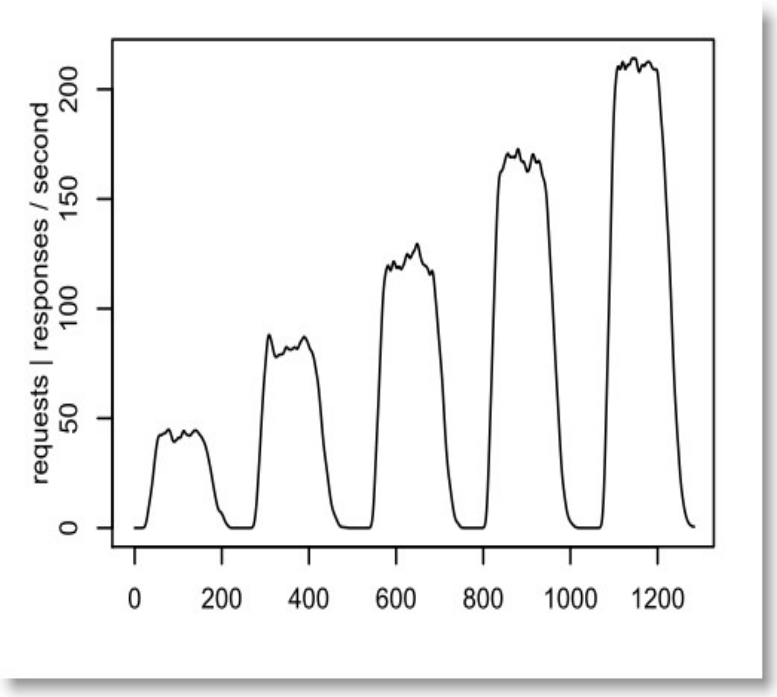

Figure 5.3:The structure of the natural load test

In this study, JMeter (JMeter contributors, n.d.) has been used to arrange for testing of the basic functions. These are:

- Control of the amount of virtual users. JMeter, in terms of its thread-monitoring; in practice, this means control of the number of simultaneous requests.
- Prevent burstness using the ramp-up procedure and by using a random-scale break between requests, and distribution of queries between the different threads.
- Generated by a predetermined HTTP calls in order to obtain a load as natural as possible.

- Combine HTTP requests and responses to them, and calculate the response time for each request-response pair.
- Saves the result in an easily readable log file for later analysis.

These requirements can be met in the test script, which is shown in Figure 5.4. Log files that are generated during the execution of a script, can be analyzed in another stage. The analysis is carried out using a high-level statistical programming language, R. Statistical analysis will be described later in this section.

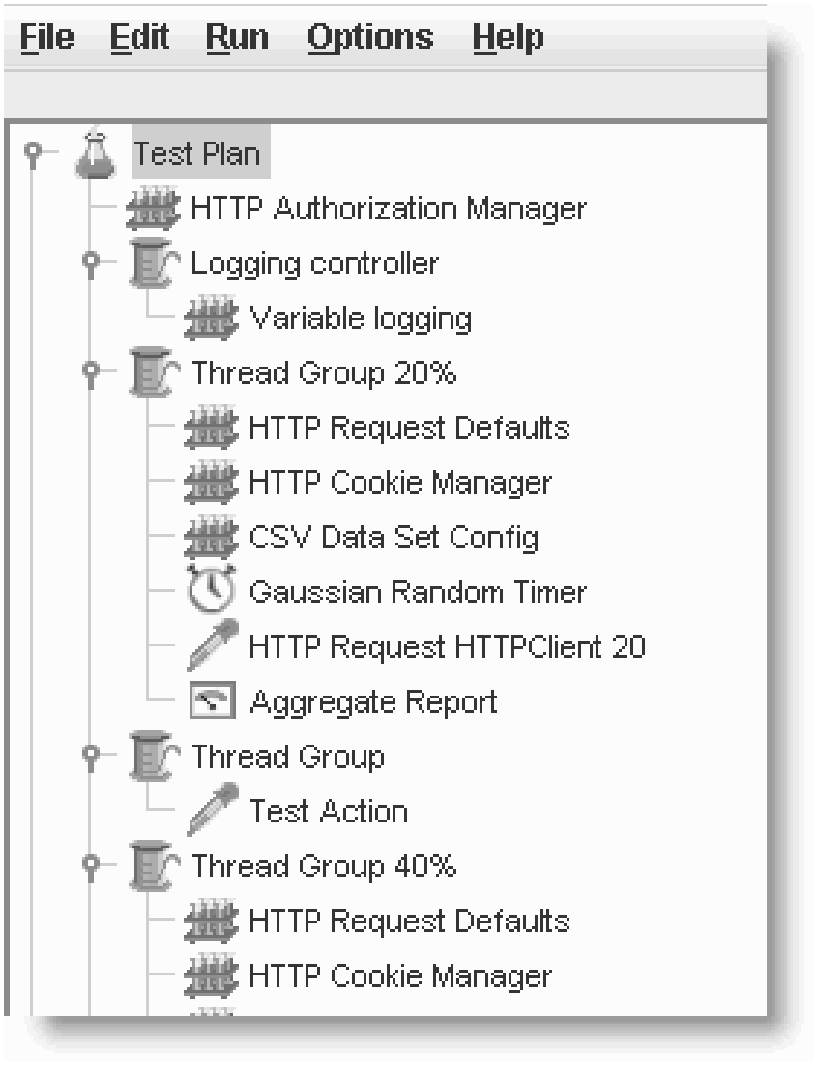

Figure 5.4: JMeter test arrangement

JMeter script structure is shown in Figure 5.4. The gradual increase in the load is shown in Figure 5.3. It was carried out using a number of consecutive thread groups. The groups are designated as "Thread Group 20%", "Thread Group 40%",

etc. Each group includes a pair of matched elements in the application. First, there are 20% of the queries to the maximum number, then the name of the group refers to the number of queries (40%, 60%. ...) with respect to the maximum number. Maximum load is generated in the "Thread Group 100%".

The load on the server will burst when the number of virtual users is increased too quickly. Therefore, at the beginning of each pulse, the load is increased slowly up to maximum number of virtual users. This is called in a basic regulation of a group as the ramp-up period. Ramp-up is the time during which the virtual users, i.e. threads, will increase from zero to the maximum in each group. If the ramp-up time is zero, all threads in the group will be launched immediately during the activation. If the ramp-up time is *T*, and the maximum number of threads is *N*, it creates a new thread *T/N* seconds. Furthermore, a too-long ramp-up time could not be used since then the desired peak load is not necessarily achieved. There may be a situation in which the previously started threads have already received a mission accomplished, and have already been completed before the last threads are activated when the desired load level is not achieved. A suitable ramp-up time is determined experimentally for each network service. This is affected by the estimated *hit rate* U_e, and the desired number of threads N_t, in which case the ideal ramp-up time is:

$$T_r = \frac{N_t}{U_e} \tag{21}$$

In addition, a random length of breaks between queries is needed to prevent burtsness. It is implemented in *Gaussian Random Timer* element. This element is responsible for the real-world time think *Z* that is described in more detail in section 3.3.2 and 3.3.4. By default, JMeter thread sends queries without interruption. In that situation, the test load can overload the server by generating too many requests in a very short period of time. At that time, the load level could be quite high, but that is not managed in terms of resource consumption. *Gaussian Random Timer* is used to create a random length of a pause before each request. Pause length is determined in the fixed part of the element and the random part. As used herein, this is the sum of the values set for a short time compared to the

conventional think time. The actual length of the pause is not relevant, because it is designed only to reduce bursts. This is due to the fact that the actual number of incoming queries is calculated from the actual log of the number of queries, instead of using the set number of virtual users, and the expected think time.

The sending of HTTP requests are processed by the JMeter *HTTP Request Default* element and the *CSV Data Set Config*. The former is the default for storage of values and the latter explains the data warehouse, where the URI addresses are stored. URI data store contains the addresses generated by the analysis of the log in the process as described above in section 4.

5.3 Resource Utilization

In this study, resource usage rate is controlled using the SNMP protocol (*Simple Network Management Protocol*) (Case et al., 1990), while the system is loaded in a known and controlled load. SNMP protocol is the result of the fact that it is accurately defined, it is generally known, it is readily available in different hardware platforms and operating systems, and may be extended. Figure 5.3 shows, how the load remains constant, in order to stabilize the controlled resource utilization. From a knowledge of the load and the consequent resource usage, the correlation can be determined.

According to *Utilization Law* (Jian, 1991) the resource utilization U_i is defined as the ratio to which a resource is in use during the time *T*. If the resource is monitoring the time *T*, and found that it is used time B_i, then the utilisation rate is:

$$U_i = \frac{B_i}{T} \tag{22}$$

If $U_i = 1$, the resource has become saturated. When a single resource has become saturated, the entire server has become saturated, and the load cannot be increased above the boundary reached by the server. If the server consists of a layer of redundant servers, the whole layer has become saturated, and all the servers are

saturated with respect to any resource on the assumption that among the servers, the load balancer is working properly. The saturation point for the entire system as a given resource is:

$$U_{tot} = \sum_{i=1}^{n} \frac{B_i}{Tn} \tag{23}$$

where n is the number of parallel servers in the system. It assumed that the servers are identical in amount of resources and the balancing of the load. If $U_{tot} = 1$, the entire system has become saturated, and the maximum throughput is achieved.

When the amount of incoming queries is changed, the change in utilization can be detected when the number of queries is sufficiently high. The background noise from external sources occurs through a valley bottom, where each pulse is followed by a short pause with no load. Load, and the correlation between the consumption of resources, is a linear or almost linear near the saturation point of the resource. Therefore, the maximum pulse peak value can be increased to 40 ... 50% of the expected saturation level without significant loss of accuracy to the result.

When avoiding the saturation of resources, it ensured at the same time that the service can be used as normal throughout the test. The accuracy of the test load increased when the load from other sources is minimized. It is done by timing the load test to take place using the service in terms of quiet time.

A virtual server allows for very high utilization rates; a fairly common commercial use of virtualization technology, VMware, has reported (VMware, n.d.) a number of customers using up to 60 ... 80% utilization rates of the server's processors. Generally, 80% utilization rate is a reasonable upper limit and 90% and should be a CPU overload warning level. Resources for each virtual machine have been allocated to start up, and since then they have adapted to the virtual machine throughout the life of the variable load. In addition, when the processes are isolated inside a virtual machine, any virtual machines does not compromise other virtual machines' resources.

Response accuracy increases when the load on its external and internal sources of the unknown proportion of the load is reduced. As shown in Figure 5.3 and Figure

5.5, the load level is about 210 rps, causing approximately 25% utilization of the application server processor's layer.

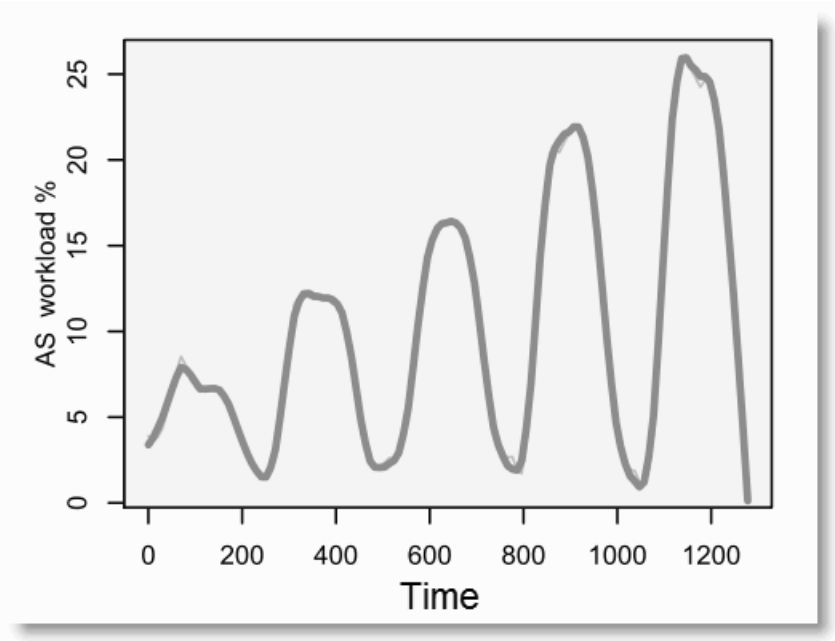

Figure 5.5:The response on the application server layer CPU utilisation during a load test

Figure 5.6 shows results for the application server processor utilization of two different test runs. In the upper figure, the sampling frequency was 10 seconds, and in the lower, three seconds. Three seconds have been selected because it is the maximum acceptable response time, for it provides at least one sample of each of the response time of the acceptance window. As noted, the results of utilization of the resource do not change significantly at a higher sampling rate. If the sampling rate is raised to a very high level, it begins to interfere with the system under test. Utilization is sufficiently an accurate understanding of the 10-second sampling intervals. The precondition is that the load is done in manageable bursts which in this study are, in principle, assumed.

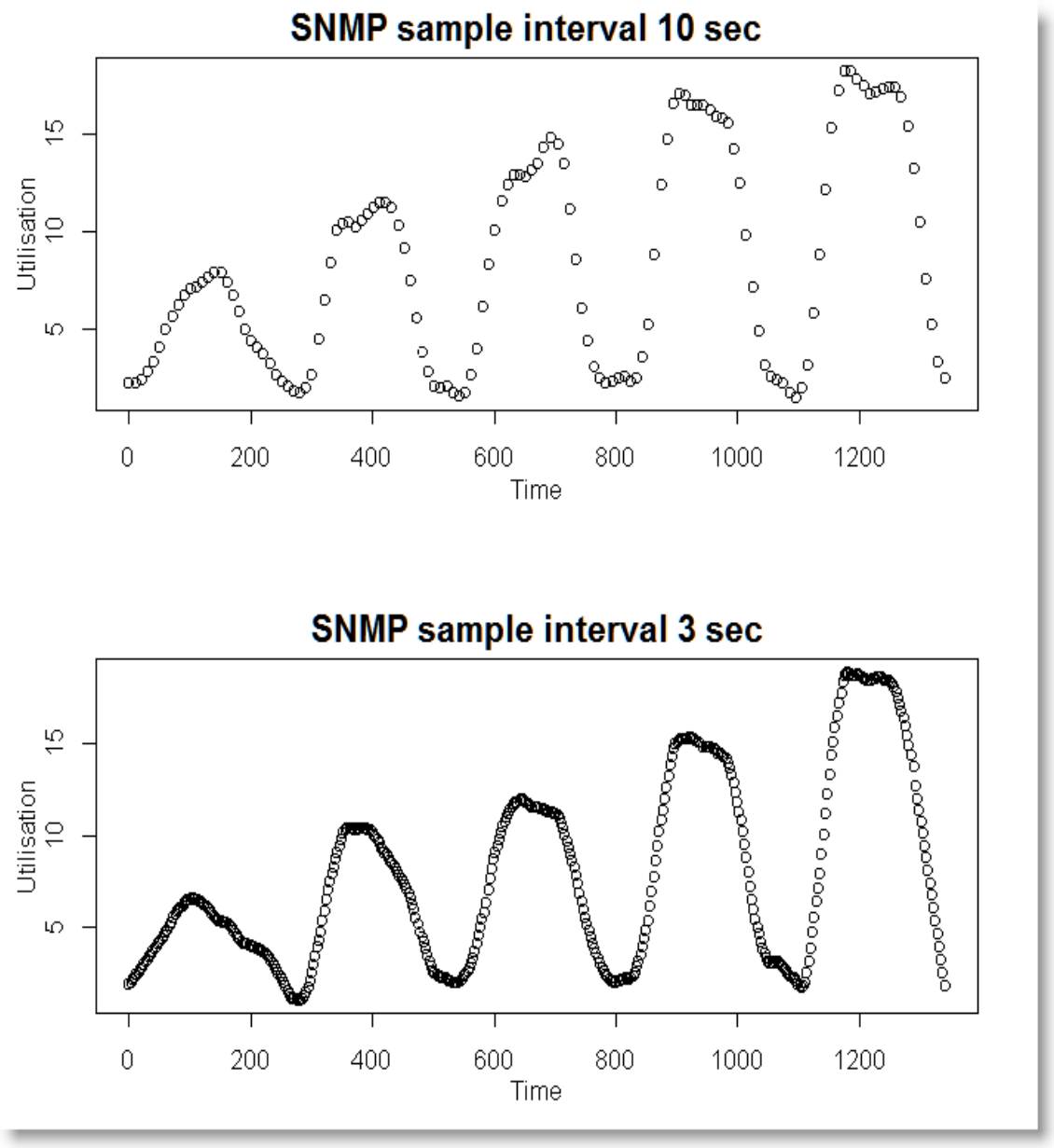

Figure 5.6: Impact of a sampling rate on resource utilization measurement results.

Figure 5.7 below shows the number of queries, response time, and the CPU load for each server layer separately. The figure contains the following server layers: a database (DB), load balancing (LB), and the application server (AS). Each load curve is the sum of all layer server resources in proportion to the available resources. Incoming request for the server is up to about 45 requests per second. This leads to more than 200 requests per SNMP query.

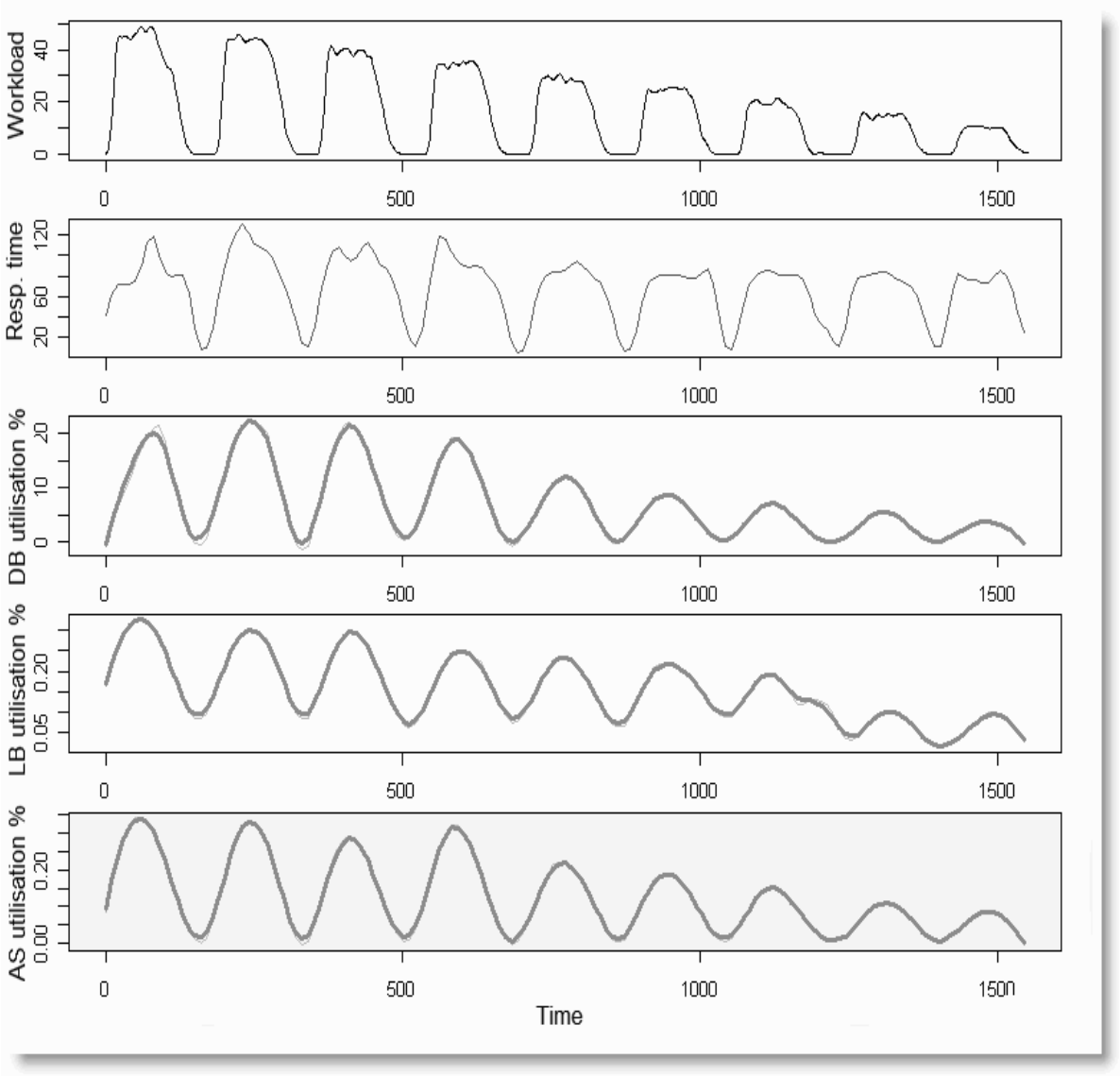

Figure 5.7: The effect of natural load test to the resource utilisation

The correlation between the resource utilization on the server and the demands on the amount of the range indicates how well the response complies with the impulse to change, as shown in Figure 5.8 below.

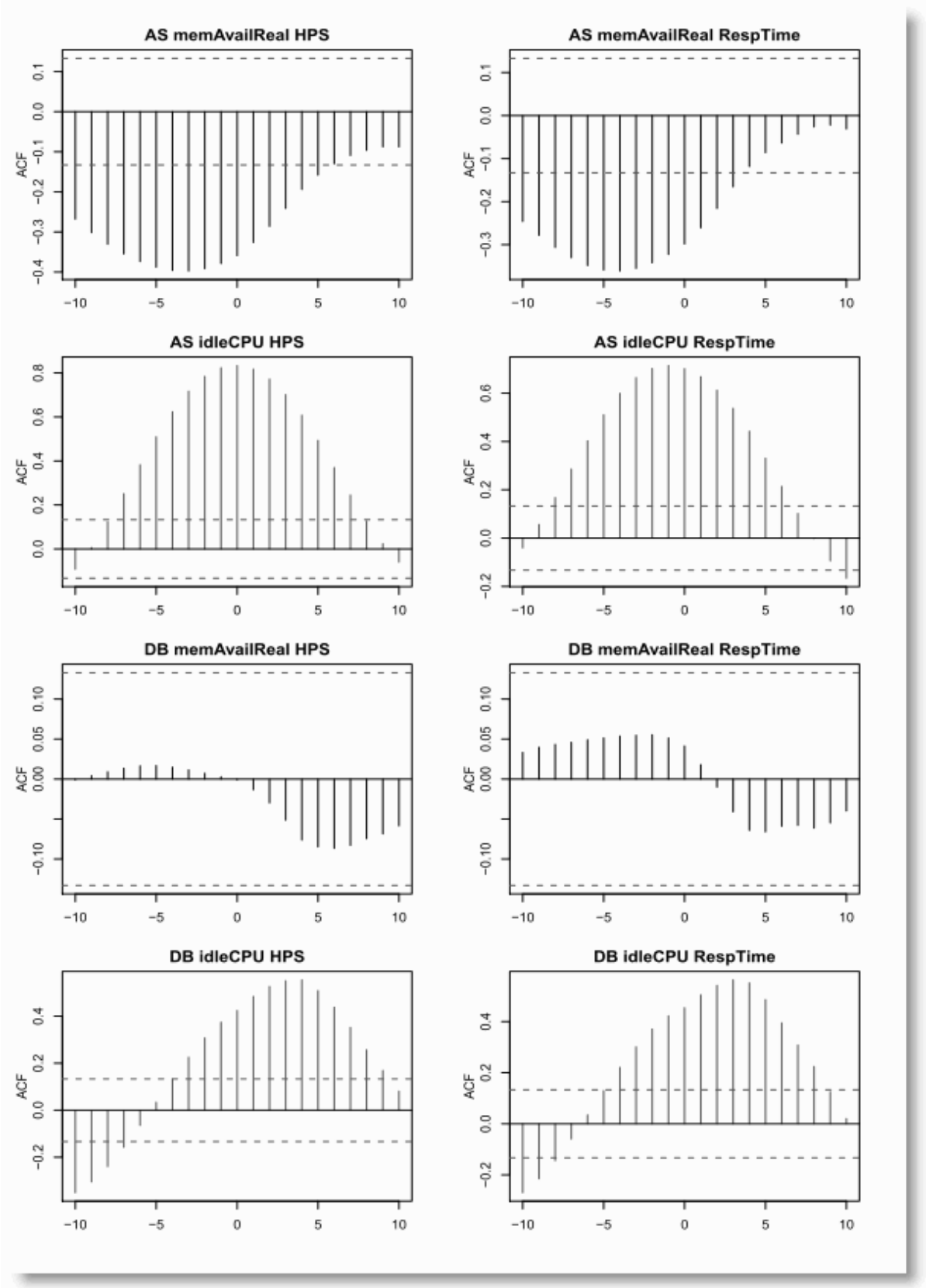

Figure 5.8: The correlation between resource utilisation and workload on left column. On the right column is correlation between resource utilisation and response time.

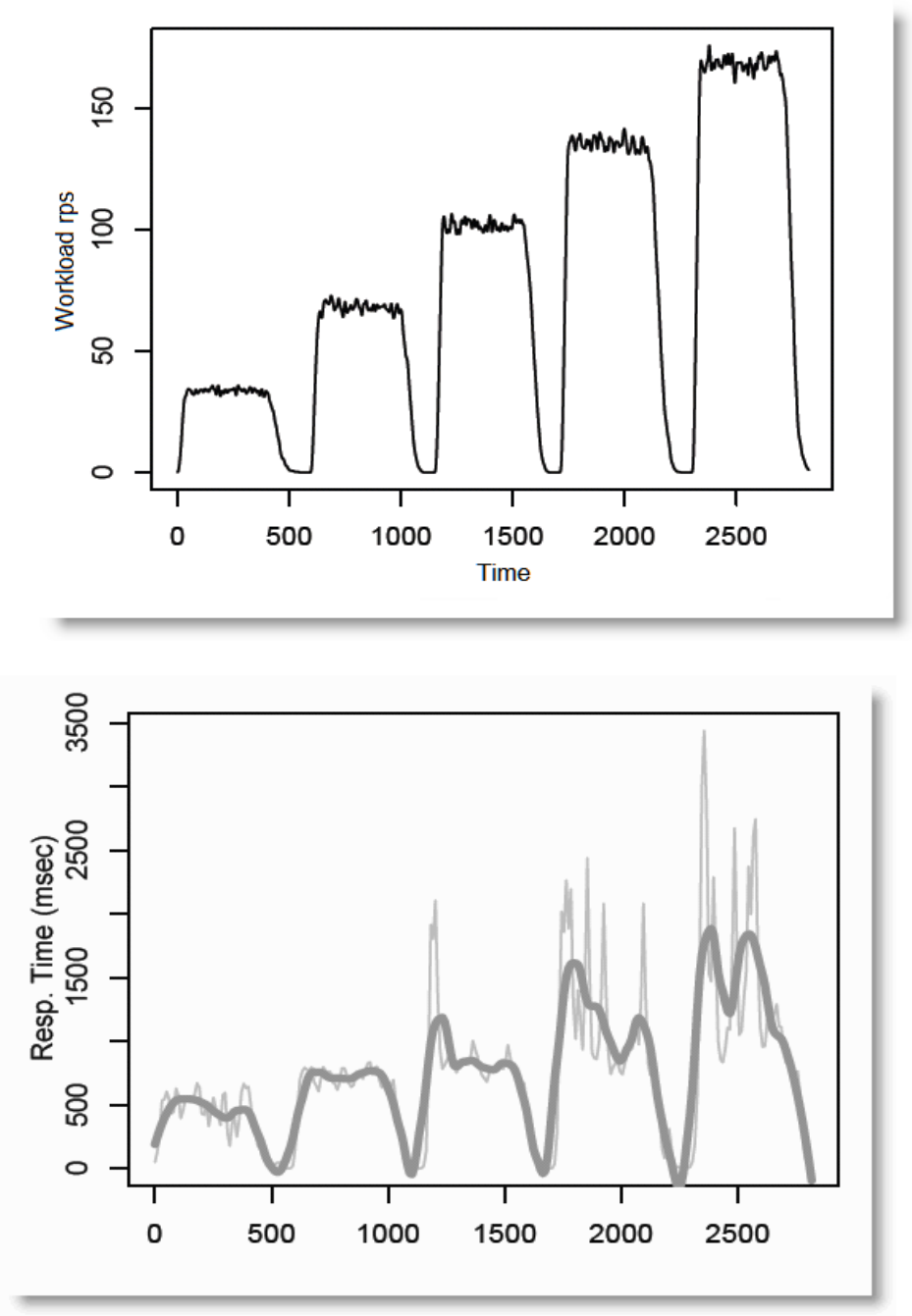

Figure 5.9:One example of the server load. Above, the increase in load causes the response time of growing up shown in the picture below, even though none of the resources is close to the saturation point.

The single value of the load can lead to a variety of utilization values. This is due to several reasons. First, the queries in the load are various so that they require different amounts of resources. On the other hand, the system's internal state may be different for different time points; this is due to the size of the backround load and the history of usage. When using SNMP to query the system status data every

ten seconds, as in Figure 5.9, the work load leads to approximately 1700 query results (170 rps * 10 sec). Since the generated combination with the URI is a very large set of queries, the simultaneous responses and results of the load are very different. Dependence between the generated load and consumption of resources can be represented by a set of points. These two variables have yet to take into account the response time.

As shown in Figure 5.9, the response time will change when the load changes. When these two parameters combined with resource utilisation are presented in mutual dependency, the result must be presented in 3D space, as it is hard to visualize in the form of 2D. Figure 5.10 shows a visualization rather than 3D-space. Facet grid can be used for all three parameters' simultaneous visualizations. The individual numerical values are not critical for the response time, which gives an idea of how close to the three-second acceptable limit is each load. On the basis of the image, it can also be seen that the set of points is arranged in a relatively linearly loading as well as the resource consumption and the response time. As response time and load increases, a set of points approaches each facets at upper-right corner. Based on the image, it can be observed that the restriction on the response time (3000 msec) is exceeded in some cases; facets (3000, 4000] and (4000,1e+05]. The number of cases in which the threshold is exceeded, however, is very small when compared with the total number of data points, n = 89 700.

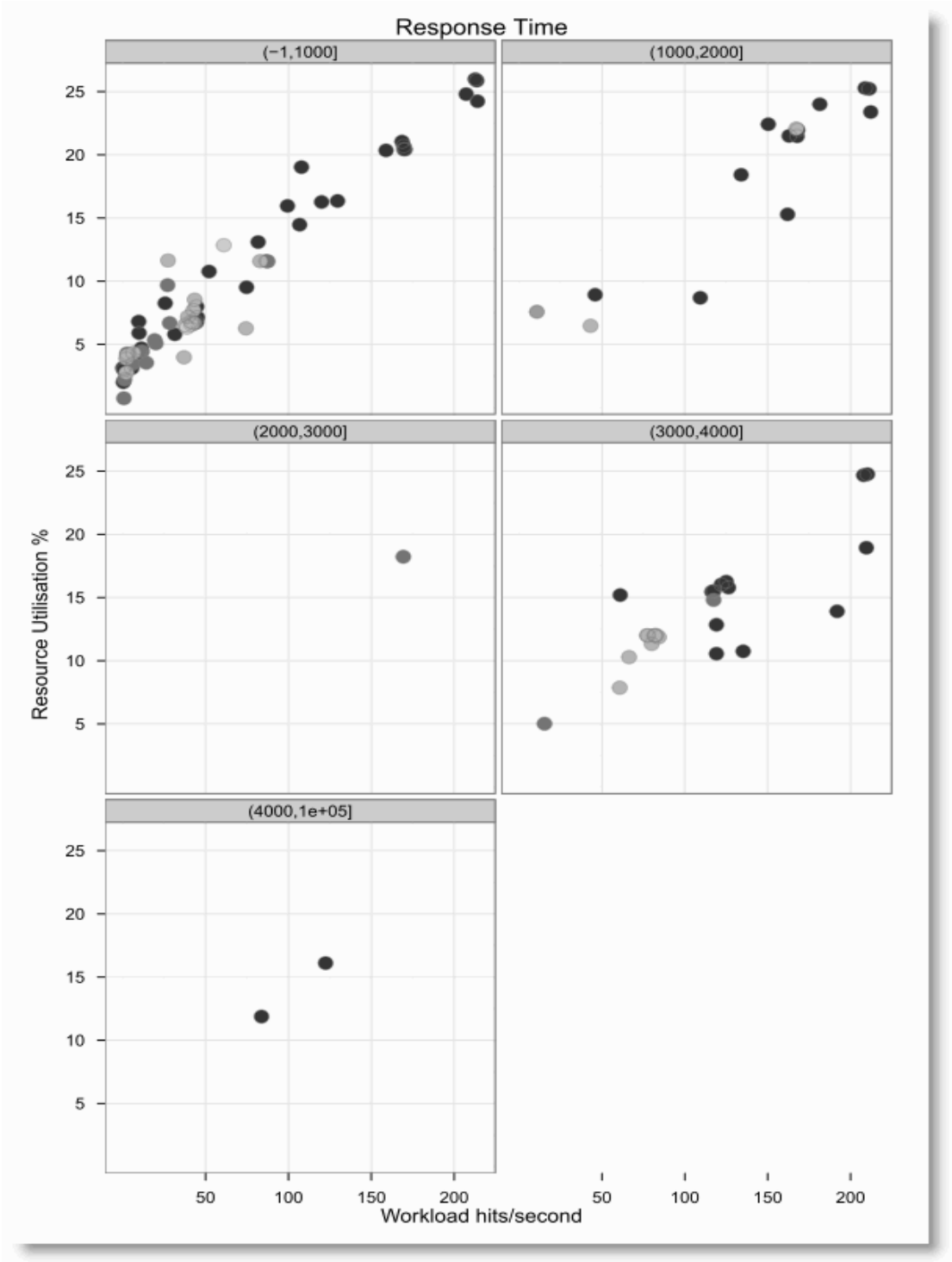

Figure 5.10: Resource utilisation, workload and response time

Another option for the interdependence of the three parameters presented describes a ternary plot which is shown in Figure 5.11. The problem is that it requires the mutual dependency of parameters, which in this case does not exist, and the result is approximate only and may support other methods. Initially, the

observation has changed to relative values between 0 ... 1. Utilization is already in the normalized form. Response time is normalized by dividing the findings by three seconds. Throughput is divided by the maximum observation values. The software package (Imai, King, & Lau, 2006, 2008) takes care of the actual data modification and presentation. In this way, detection data has been modified to a shape in order to determine the balance between the variables, and the closer the triangular centre of the observation points, the better is the findings in balance. Figure 5.11 shows that the response time of observations has exceeded the permissible three seconds, and the workload is thus relatively low. Workload may be sufficient to find that the system is loaded with a variety of mixed loads.

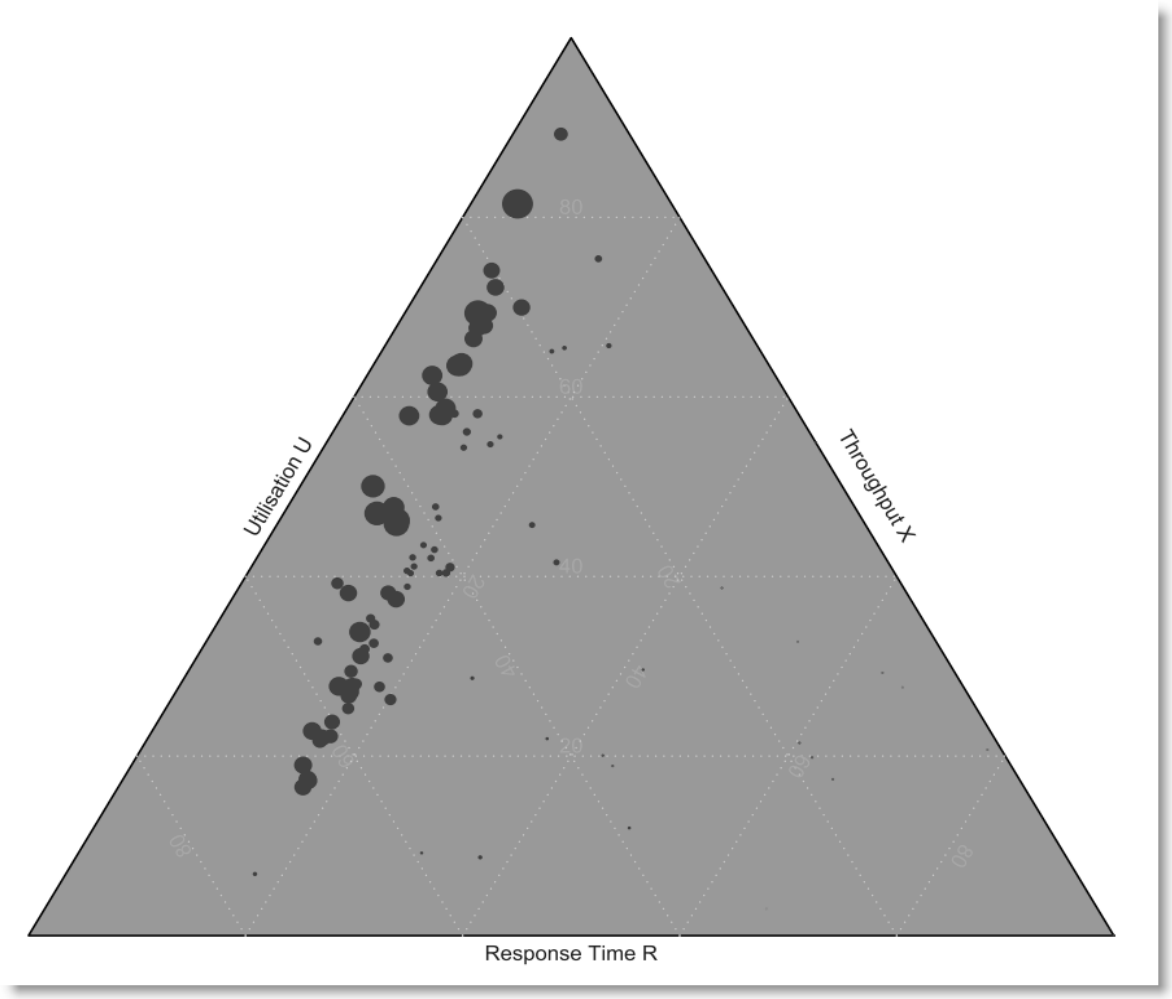

Figure 5.11: Ternary presentation of utilisation, workload and response time

The website user can detect the response time exceeded without none of the resources utilization limits being exceeded. The system may be configured so that,

for example, it limits the number of concurrent connections (Apache max_conn), or the number of simultaneous sessions (Apache max_sess). If such limitations can be detected using SNMP queries, the method described above can be used to determine the system saturation point. However, it is unusual that such programmatic boundaries cannot be found using SNMP queries. The system must therefore be loaded at least once in such a large load that such restrictions will become apparent.

The correlation between response time and resource consumption shall be examined in all cases where programmatic constraints are found. The most limiting factor (*MRF*) is obtained as follows:

$$MRF = min(maxload_{utilisation}, maxload_{response\,time}) \quad (24)$$

As explained in Section 2.5 above, the system performance is not constant over time, it will change as a result of the phenomenon of *software aging*. Furthermore, as explained in Section 3.3.4, the system has always one or more bottlenecks. A carefully designed equipment is one in which the various resources are in balance nearly or completely, in which each resource is not significantly more utilised than others, and the system is optimized in terms of costs. In this case, without further analysis of the performance, bottlenecks cannot be detected in the system. Far from the equilibrium point of the system is always a performance-limiting resource that is already identified in previous analyses. In practice, the service cannot be completely balanced; yet always, one of the server layers and a single resource group is the bottleneck.

5.4 Sensitivity Analysis

The load test is carried out on some of the outcomes of the factors affecting accuracy. This section seeks to identify those most significant to assess their impact on the outcome and to assess the likelihood of their occurrence, if relevant.

First of all, the ration between load used in the testing and the assumed or calculated performance. Incoming requests and the arising loads cause the

consumption of resources which is linear up to the knee point. At the knee point and above, the consumption of a resource begins to grow vigorously, and operation of the system becomes unstable. It is therefore appropriate that the service operates continuously below the knee point, and also during load testing. Operating in the linear part of the test loading conditions of the target value may be allowed a fairly large fluctuation. In this study, the guide value used is 40 ... 50% of the maximum value. Experience has shown that the results do not radically alter the range 20 ... 80% of the highest value.

Second, resource consumption is also affected by the number of requests coming from outside of the test. Their significance may be assumed to be equal to or greater than that of the test load requests. Their importance cannot be completely eliminated; the only way is to schedule a test load in such a way that they account for as low as possible.

Thirdly, the server environment itself gives rise to a variety of contingency factors. These include, for example, the operating system, the metering (here, the SNMP), the database, the application server, etc. Their origin and the differing versions of these products cause various kinds of uncertainties. This study has used two of the most frequently appearing operating systems (Linux and Windows), application servers (Apache and IIS) and databases (MySQL and SQL Server). These results can be judged only by comparing the representation of time series. The results obtained do not give rise to the need for a further review.

Fourth, the metering accuracy, i.e. control of resources that are relevant to the service load capacity. If the bottleneck in the resource is not involved in the control, it will be apparent, therefore, that the response time is exceeded, even though none of the resources to be monitored is saturated. In this case, control of resources must be added or clarified. The method does not give direct information about what resources are needed to be monitored more closely.

5.5 Conclusion

In this chapter, the system under test to a load is presented in such a way that it is well-known, both quantitatively and qualitatively. The chapter showed the loading system to explore the different sizes of load impulses, and to induce change in the server resource loading rate. In addition, the utilization of resources varies, while the response time changes as the load changes.

By observing the load rate and the consequent change in the use of resources, the bottleneck of the whole system can be identified which further allows for the performance of the system. In the next section, a performance analysis is made on the basis of the collected data.

6 Performance Analysis of the System

This chapter provides an analysis of the data, collected on the basis of the previous track. A result of this analysis is the performance prediction in different load rates of the system. The chapter goes through four different network workload results over a 90-day period. The network architecture is shown in Figure 5.1.

In examining the performance of computer systems, there are several commonly used gauges. The best-known is the response time (also known as speed, turnaround time, reaction time), throughput (sometimes also called capacity or bandwidth), utilization, reliability and cost, or performance ratio. The two most interesting gauges on e-business are cost and throughput. In addition, reliability is important. Throughput can be improved by investing in equipment or software or both, but the costs are growing rapidly as discussed in Chapter 1. The most important indicators of the cost optimization point of view are: request arrival rate, throughput, and response time. The only infrastructure resulting indicators are response time and throughput. This is due to the overall performance of the servers, which in turn can affect the tuning of applications and configuration. The maximum acceptable response time is set by management.

The chapter validates the reliability of the forecasts produced by the model. Validation of the problem has two aspects. First, the validity of an experimental model, i.e. the computational structure has to fulfill the intended functions and logical structures. Second, how well does the experimental model represent a real system, and how far does it behave like a real model. The first approach is easier to be satisfied.

Each HTTP request response time is determined separately. Therefore, there is no single clear response time, but it must use an average response time which describes all the HTTP queries performed between the SNMP queries. Response time indicators describing the properties have been discussed in other studies (Ciemiewicz, 2001; Mielke, 2006).

6.1 Precision and Accuracy

Website performance characterized by sequential measurements does not produce exactly the same results. Instead, there is an area in which the results are placed. Figure 6.1 is used to illustrate the difference between the measurement accuracy and precision. Figure 6.1(a) illustrates the precision. The measuring may include a systematic error, in which case the values are inaccurate but the variation is small. Figure 6.1(b) illustrates the measurement of the number resulting from correct values, but the variation is large when the method of measurement is inaccurate. Figure 6.1(c) illustrates the target state with a good precision, and the deviation is small when the accuracy is good.

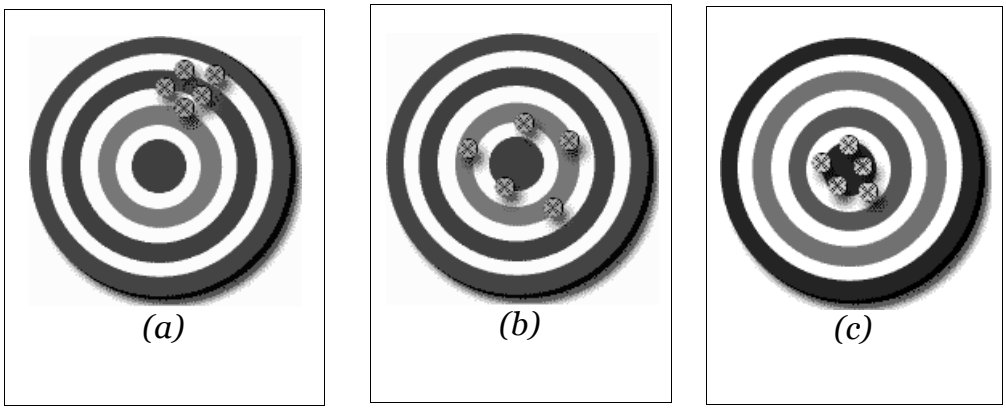

Figure 6.1: Precision versus accuracy. In (a) low accuracy is presented with high precision and (b) presents a high accuracy with low precision. In (c), there is a high accuracy with high precision.

Although the analysis is performed automatically, it still seeks both accuracy and precision as much as possible. The problem, however, is that the evaluation of the source material is only available for programmatic resources. This can easily lead to a very complex programmatic interpretation analysis of the appropriate data to be formulated alternatively, so that the relevant data has to be rejected if it contains the incompatible characteristics of the rules. In this study, the simpler inference rules have been chosen, which will have to accept the risk that the relevant information will be rejected. Supposedly, however, in the longer term, the problem can be compensated when an increasing number of analyses will be carried out.

6.2 Data Preprocessing

Data preprocessing has three objectives. First of all, the fact that the individual random fluctuation becomes less important. Secondly, the findings are of a different sampling rate, which is aligned prior to analysis. Thirdly, the relative proportion of the findings; they are changed into a usable form prior to use.

The individual random fluctuations can be reached in a very wide variety of sources. Mindful that the most common cause is a server test apparatus and the logical distance between the data communication networks, this is what causes the fluctuation in performance of the data communications. The server hardware resource usage may have short-term spikes due to natural causes. These situations include a service request from an external source, which will require a lot of resources. Spikes may also be negative, i.e. a constant load drop suddenly for no apparent reason. Such might occur, for example, in short-term communications outage. These random variations may be displayed in the monitoring equipment, but they are not covered by the web server system in normal operation; they interfere with the analysis, and thus their removal is essential.

SNMP queries are to take place at regular intervals, typically every few seconds. Natural variability may arise from carrying out the survey equipment congestion. At the same time, the SNMP queries occur with the URI queries; their intermediate may be a few milliseconds. These events are made in time into a form that makes it possible to perform the analysis. Since the interval of SNMP-query is the coarsest available unit of measurement, it must be adapted to a URI query interval for the interval of SNMP queries. Each URI query also has its own response time, when the term of all queries and response time shall be summarized such into a statistical indicator that can be used to describe the response time as accurately as possible.

Many of the responses of SNMP queries are in such a format that they cannot be used as such. One such response is the current amount of memory available. Alone, it is not relevant; in addition, we must also know the total amount of memory. Furthermore, the CPU load ratio is into a similar situation. Some of SNMP implementations are such that they return the CPU number of events from the

previous server startup time. Another possibility is that the number of events is obtained as the sum of various events. The actual load on the CPU is then:

$$CPU_{tot} = CPU_{user} + CPU_{kernel} + CPU_{wait} \quad (25)$$

It is also necessary to know how many processors such as events are created. Processors or the number of threads and total amount of memory is usually obtained via SNMP query. It should be noted that the different test runs at hardware resources may be different. In particular, the use of virtual hardware, new resources, increase or decrease, is technically very easy and does not require a service interruption. Thus, for each test run, it should be necessary to upgrade the hardware information for data normalization. In this study, it has been assumed that the total amount of hardware resources will not change during the test.

6.3 Validation of Measurements

Authentic queries can be carried out to validate the load so that the loading rate is increased until the saturation point is reached or an acceptable response time limit is exceeded. In this way, the measured maximum performance should be the same as the analysis result obtained. The model does not produce exactly the same results load, because the load is caused by queries and the system's internal state is not the same for each recording session. However, the accuracy can be increased by repeating the tests several times within a short period of time.

The analysis results will be independent of the load used during the loading rate, so that the loading rate and tested resource utilization should be found between an adequate correlation. Figure 6.2 shows the impact of loading rate on the forecast of maximum performance. As we can see, throughput does not depend on load rate to be used. Throughput is mainly located on two different levels of approximately 400 ... 450 rps and 900 ... 1200 rps regardless of the load which varies between 20 ... 90 rps. Findings in the layers are consistent with Figure 6.12. It thus appears that for the test used in a load factor, there is little effect on the resulting performance of the forecast.

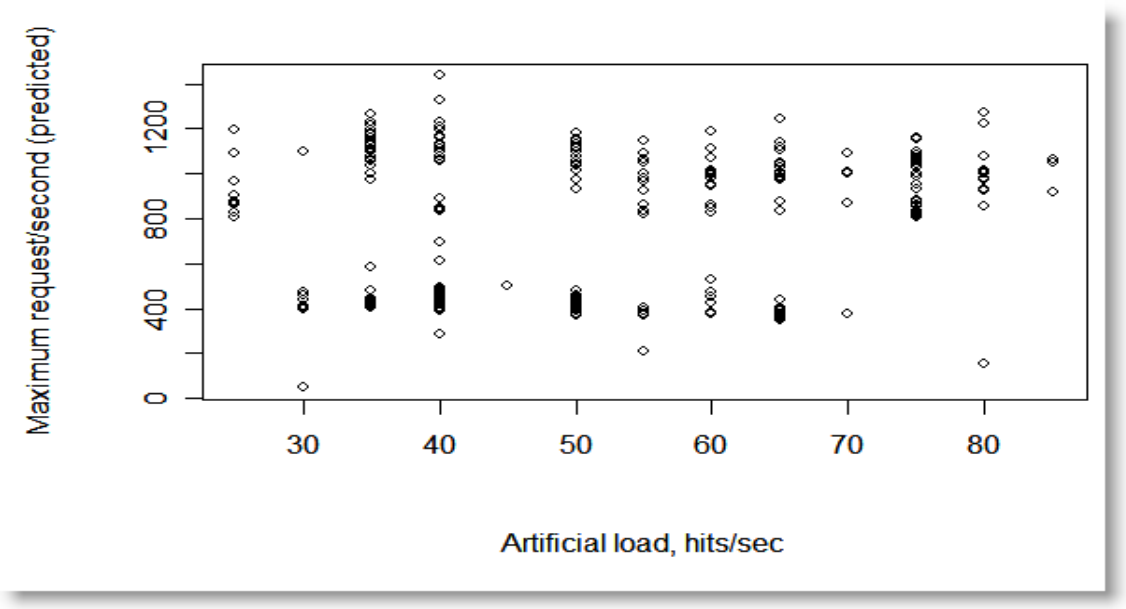

Figure 6.2: The effect of load test rate for the predicted performance in website A.

Load and resource utilization formed by the observation group hit rates and standard deviation can be described by using the *adjusted* R^2 value (Cameron & Windmeijer 1997). A set of points are combined by means of linear regression, and the suitability of the result is described using the value of R^2. When the regression line obtained extends outside the region measured by using the linear extrapolation, it can be found in the performance forecast for the server to a specific resource. At the time of measurement, it is the most limiting resource in the current bottleneck in the system. This result is sufficiently accurate in determining the overall system performance.

6.4 Interdependence of Measurable Factors

Resource consumption, but also response time, plays an important role in assessing the maximum performance of a system. At the same time, as the load increases, it generally increases the response time. Increase in the response time is usually due to the fact that resources have to wait. The general increase in activities for the server hardware system means an increase in almost all the events in terms of resources such as CPU, memory, disk I/O, network traffic, etc. In Figure 5.9, the

load increases from 40 rps to 170 rps, and response time will increase from 500 ms to 1700 ms. However, none of the service resources is close to the saturation point. Therefore, each of the analyses shall consider the response time of the estimate. The response time applies to the entire system, it is not resource-specific.

Figures 6.4 to 6.7 show the measured response time (*R*), the number of incoming requests, and resource utilization (U_i) of key resources in one measurement run. Resources utilization is collected for each resource or resource group separately. The total system throughput and response time are relevant only to the whole system level. In this study, the resource-specific throughput or resource-specific response time is not relevant. As we can see, the equations (3) and (4), over a sufficiently long time arrival rate, should be equal to throughput, $\lambda = X$. Therefore, the images have been used instead of the arrival rate or the term throughput.

The arithmetic mean to describe the use of response time has been discussed in another study (Ciemiewicz, 2001). It focuses on outlier due to biased results, especially in terms of the SLA agreement. The study shows how even a single outlier can distort the value of response times describing the key figure. However, it proposes using either the geometric mean or the median. Table 5 shows website A1 loading test response times, calculated by various methods. Individual response times are converted to 10-second intervals so that the response time period is represented by maximum, median, arithmetic mean or geometric mean.

Table 5: Summary of response time values on website A1

	Max	*Median*	*Arithm. Mean*	*Geom. Mean*
Min	0,0	0,0	0,0	0,0
1st Qu.	39,0	9,0	11,7	10,7
Median	743,0	11,0	19,8	12,3
Arith. Mean	1 086,0	9,4	27,9	11,8
3rd Qu.	1 462,0	12,0	37,3	15,9
Max	10 070,0	31,5	114,9	30,1

All four of the table methods used to correlate well with the load level increase. For the service user, it is important that the response time remains below the desired

limit value. When the load factor rises enough, a fraction of the total response time exceeds the limit value. Distribution of the response time should therefore be inclined to the right (positively skewed distribution). In addition, tail should be as short as possible, and then the number of long response times would be minimized.

Only the long response times are interesting in this analysis. The Table 5 shows that when the data is formed by the maximum values, the long response time will remain. Figure 6.3 shows the maximum values of the calculated density plot, which is found to be strongly skewed to the right. The figure also shows the 3 000 ms threshold value by a vertical line. We note that the threshold exceeds to some extent. They are also reflected in Figure 6.4 to Figure 6.7 - Response Time vs. Throughput graph. It is obvious that the response times may be processed only using a maximum value. The problem is the higher variability than other methods; on the other hand, it is the only one that preserves the necessary information about the long response times.

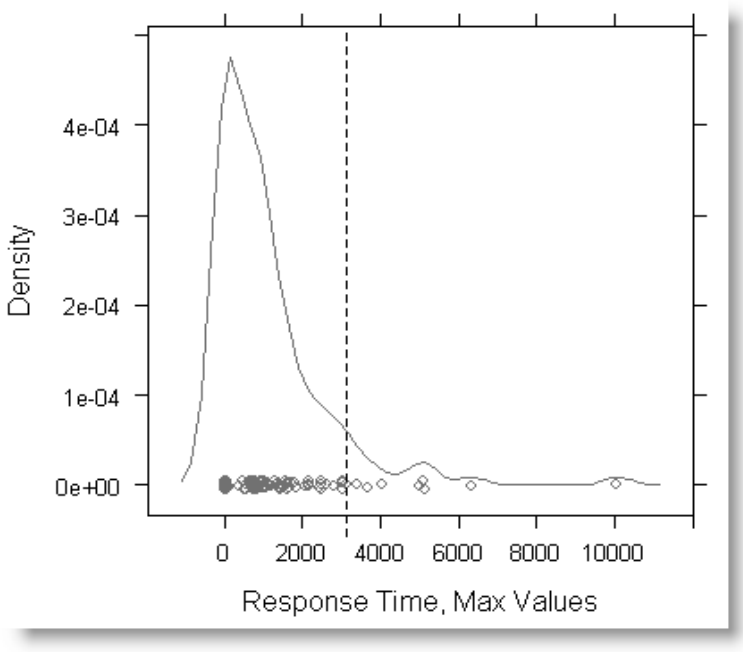

Figure 6.3: Density plot of response time values in the application server of website A1.

Figures 6.4 to 6.7 show that the material is formed from four different sites. In three cases, the most loaded server has been the application server: Figure 6.4, Figure 6.6, and Figure 6.7, and in one case, the database server, Figure 6.5. Each

service is located on a different hardware, different application and different content. In addition, services are aimed at different segments of the public. Data is summarized and the resource utilization is collected using SNMP query every 10 seconds. The period includes the response time and mean of throughput. The arithmetic mean is selected to describe the data by the fact that the various peaks do not intentionally affect eliminated. The obtained data is illustrated in Figures 6.4 to 6.7 so that the connection between the results forms a set of points, and the points are connected to 3 degree spline.

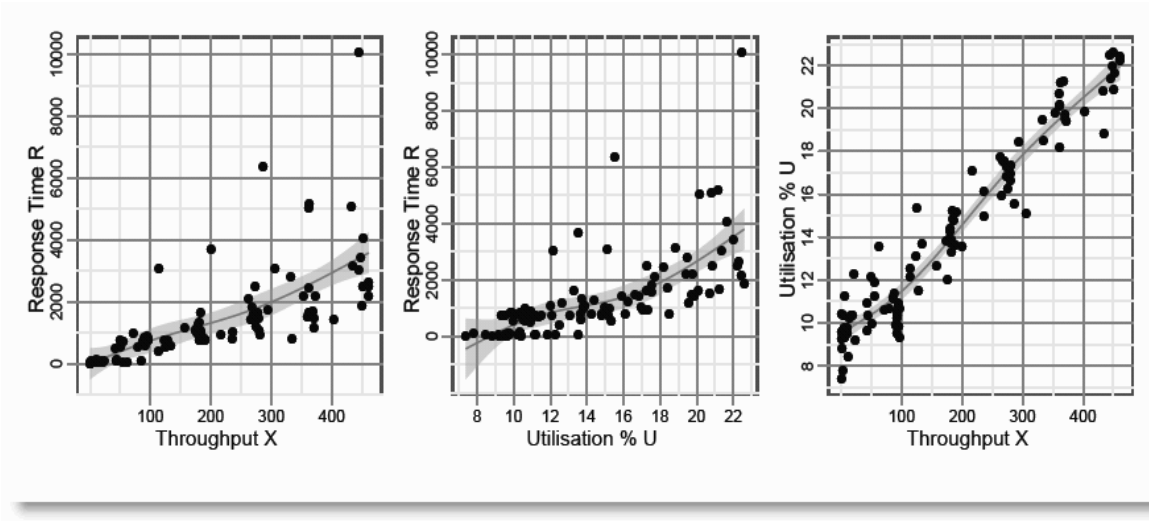

Figure 6.4: Measured throughput (X), response time (R) and utilisation % (U) in application server on a type A website A1.

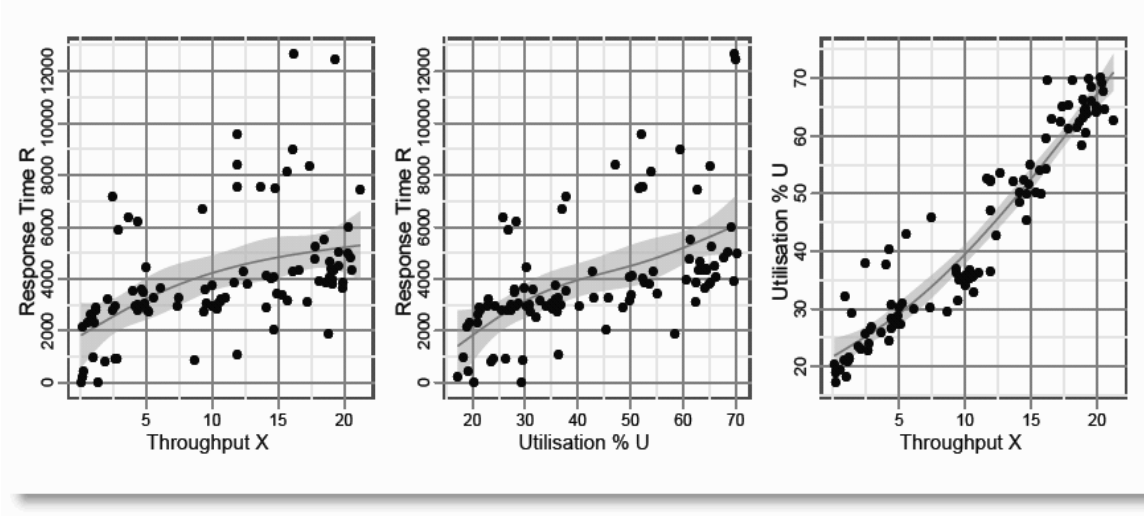

Figure 6.5: Measured throughput (X), response time (R) and utilisation % (U) in database server on a type B website B1.

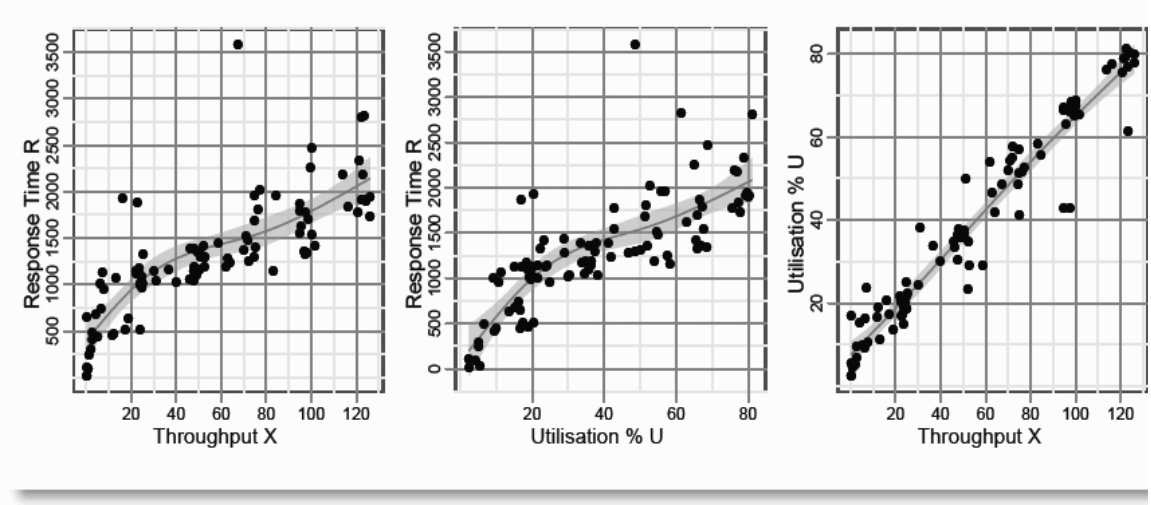

Figure 6.6: Measured throughput (X), response time (R) and utilisation % (U) in application server on a type B website B2.

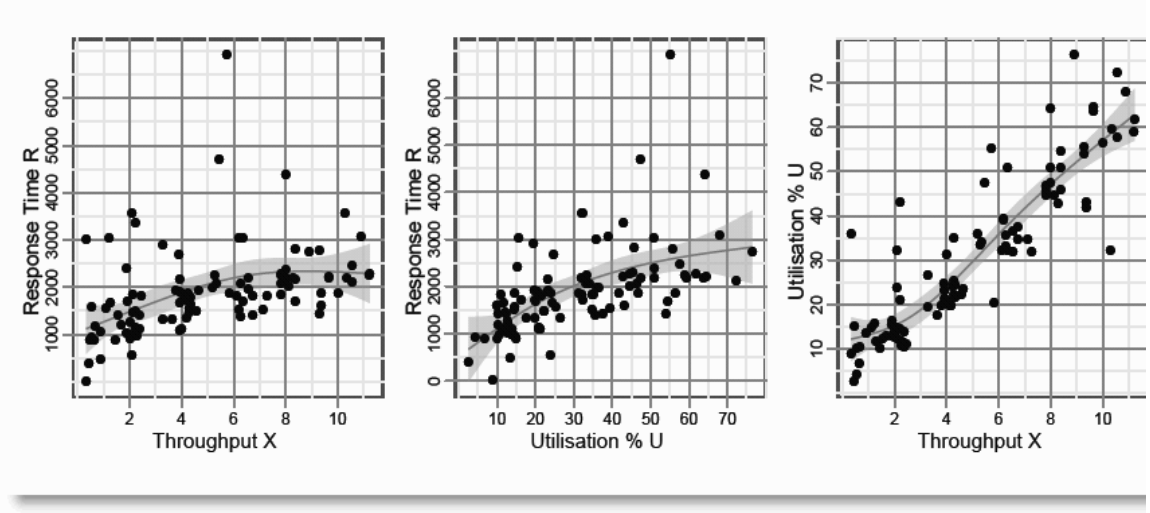

Figure 6.7: Measured throughput (X), response time (R) and utilisation % (U) in application server on a type B website B3.

As noted, the connection between the throughput and the response time is not as that assumed in Figure 3.5, as in any measured case. The measured results are from the network systems, each containing a number of servers. Therefore, it is possible that a low load, whose effect is different from the previous studies, has been demonstrated. Communication between the servers takes time and using high-level application solutions for various pools and protocol operation is not optimized for speed but in terms of usability.

In Figures 6.5 and 6.6, two different service load performance are presented; the first is the database server, and the second is the application server CPU load. As

the figures clearly show, when the load increases, the response time will increase more sharply at the beginning, and then stabilized. For the server processor, each of the load rates is very high, being very close to the saturation point of the curve and is due to sharp returns at high throughput values. If the load continues to increase, as well as response time, utilization appears to be referred to a very sharp rise. If the load continues to increase as well as response time, utilization appears to be referred to a very sharp rise.

Dependency is linear or almost linear between throughput and utilization rate, at least until 80% utilization rate, as shown in Figure 6.6. This result reinforces the idea that the processor capacity can be safely used for about 70%. It is possible that an available capacity as shown in the Figure 3.9 is higher than the limit shown, and in particular, the lower limit saturation is higher. The difference may be due to the fact that as hardware advances, the border of available capacity will increase.

6.5 Performance Prediction

Figures 6.4 to 6.7 set out the resource groups for a maximum out-performance forecast using a linear extrapolation. It will be extended by linear extrapolation to forecast maximum accepted response time up to three seconds. Similarly, the connection between utilization and throughput capacity will be expanded by the linear extrapolation until the resource maximum allowable rates of utilization. Thus, the results obtained, when selected, give the minimum performance of the system.

Figure 6.7 presents the test result formed by means of linear extrapolation forecast of the performance of the application server, which is shown in Figure 6.8. Since the resource utilization is far from saturation point, linear prediction is well suited, such as shown in Figures 6.4 to 6.7. In Figure 6.8, the maximum resource performance value is 2150 rps when the utilization rate is 70%.

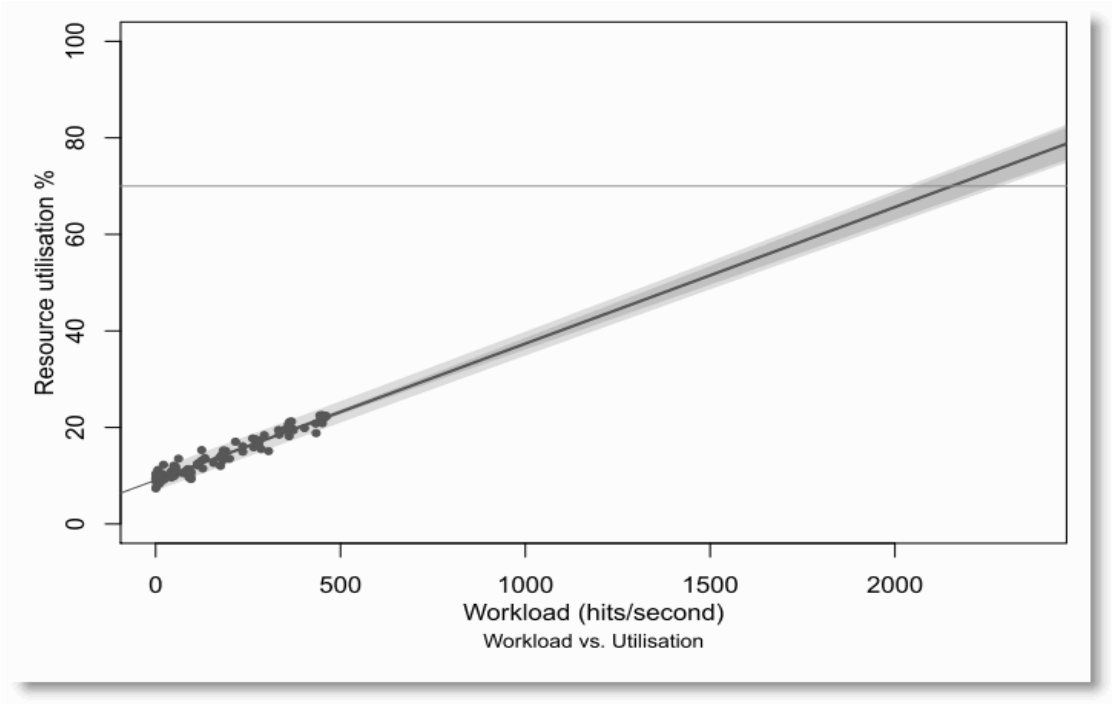

Figure 6.8: Estimated maximum throughput in application server on the website B3.

The same tests as in Figures 6.4 - 6.7 are shown; Table 6 shows the maximum performance predictions for other resources. Because extrapolation is necessary to consider a maximum value as well as sensing the linear curve fit to values in the table, the R^2 value is also shown. The forecast accuracy is not enough to get an accurate picture of just data points describing the curve. Therefore, the maximum performance in the table is shown precisely in the curve fit rather than by means of a simple minimum value.

As we can see from the table that the linear extrapolation not always gives reasonable results, in some cases, it gives very low R^2 values. These have been omitted from the table. Generally, the phenomenon is due to the fact that the server resource consumption of the layer is so small that it does not sufficiently correlate with the load increase rate. In this case, it is clear that the resource cannot be a bottleneck in the system.

If the limiting factor in the analysis is response time, the real bottleneck in the test has not been found; in other words, a critical resource load data was not collected using SNMP. Exceeding the allowable response time always indicates a lack of

resource, because it cannot in itself be a bottleneck. When the response time threshold is exceeded, the resource monitoring needs to be extended.

Table 6: Summary of throughput values within one test run at websites A1, B1, B2, and B3

Resource i	***CPU Utilisation***		***Memory***	
	Estimated	R^2	*Estimated*	R^2
Website A1				
Load Balancer	24 400	0,599	589 000	0,037
Application Server	**2 150**	0,936	987	0,246
Database Server	-	-	-	-
Website B1				
Application Server	-	-	-	-
Database Server	**22**	0,9	-	-
Website B2				
Application Server	**109**	0,941	-	-
Database Server	4 140	0,213	44 500	0,091
Website B3				
Application Server	**13**	0,795	45	0,881
Database Server	112	0,635	-	-

When approaching a saturation point, it shows a strong increase in resource utilization as the load increases. In this case, the extrapolation will result in excessively high-performance values. One has to use the Weibull growth curve equation rather than the linear extrapolation:

$$U = a - be^{-c\lambda^d} \tag{26}$$

, where a= horizontal asymptote on the right, b=difference between the asymptote and the intercept (the value of U at $\lambda = 0$), c=the natural logarithm of the rate constant, and d= the power to which x is raised.

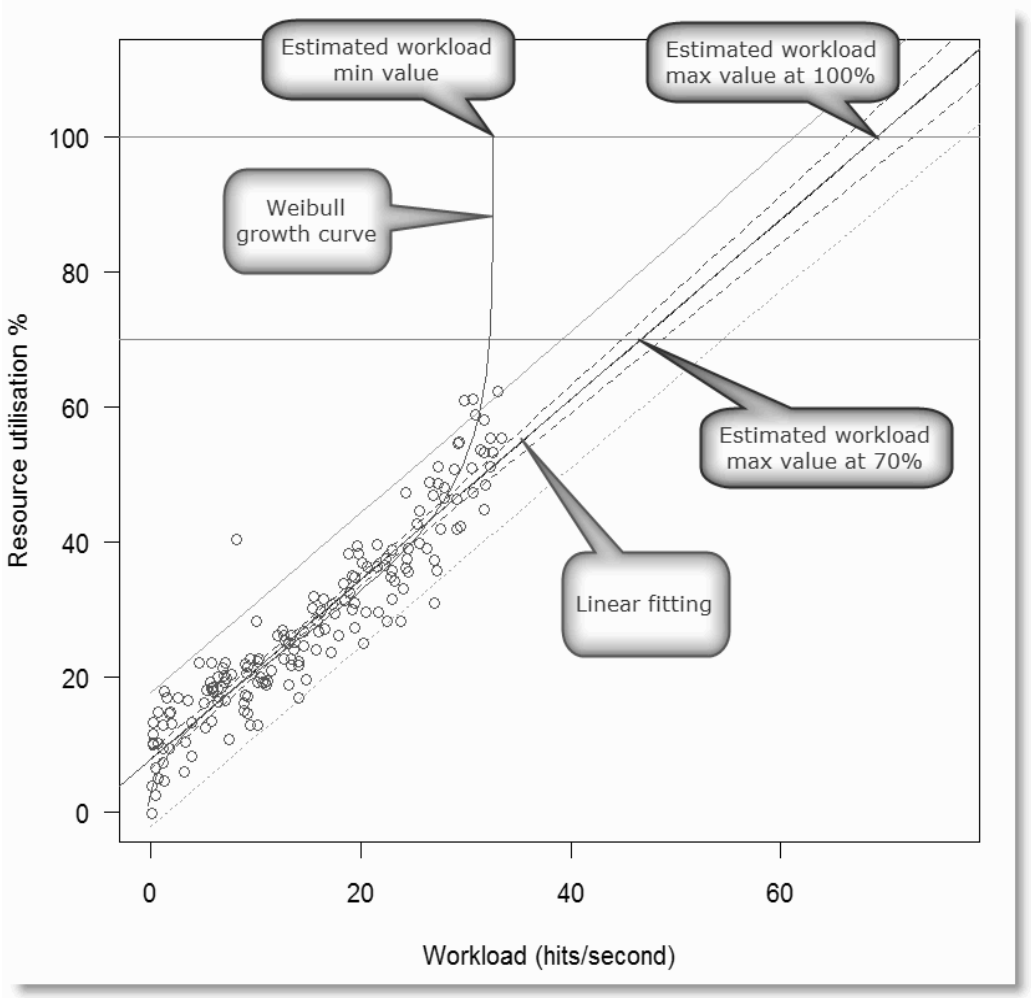

Figure 6.9: Fitting of measured resource utilisation using different rates of natural workload

Figure 6.9 compares the linear extrapolation and the Weibull growth curve. The maximum value of throughput is derived from the linear extrapolation and the growth of the Weibull growth curve. When using the linear extrapolation, the resource 100% duty cycle is the maximum throughput of 69 rps and the *adjusted* R^2 value is then 0.873.

There are very few studies on acceptable resource saturation levels. Maximum utilization of the resource depends on the type of resource. For processors, it is typically 60 … 70% respectively, the memory utilization rate depends on the total amount of memory, operating system, and the amount of memory needed by the application. When using a 70% utilization rate, as shown in Figure 6.9, the predicted throughput is 47 rps. As a kind of reference value, memory consumption can probably be considered to have a maximum value of 90%.

The different methods of calculating the values produced and their relevance indicators and *adjusted* R^2 values are summarized in Table 7.

Table 7: Forecast accuracy using different calculation methods

Method	*Estimated performance*	*Adjusted* R^2	R^2
Linear 100%	69	0.87	0.87
Linear 70%	47	0.87	0.87
Weibull	33	0.89	0.89

As seen from the Table 7, a set of points fits quite well with all the models used in the calculation, both R^2 and *adjusted* R^2 are reasonably close to one. The table confirms the understanding formed on the basis of Figure 6.9 that the linear model produces a performance of over-optimistic results; in particular, using a of 100% saturation rate. On the other hand, the resource utilization is low, as shown in Figure 6.8. Weibull growth curve produces too pessimistic results. This study has concluded that if the resource utilization is not more than 50%, a linear model is always used, unless it is used for Weibull growth curve.

6.6 Visualization of Results

When the load test is repeated several times on successive days, there will be a time series of performance of each individual resource. According to a website resource, such a performance time series is shown in Figure 6.10. The measured values are summarized for a single day in Table 6. In Figure 6.10 the measurement time is a green vertical line, the left side of the values are the measured observations, and the straight lines on the right side are the predictions of performance. The line with the same color wide area represents the 95 % confidence interval. It is normal that the predicted performance loadbalancer range is large, because its load is small compared to the other server's layers. In practice, the most potential bottleneck is always a best scenario; that is, standard deviation of measurement results is the

smallest, and the confidence interval is the narrowest. As with Table 8 and Figure 6.10, the most limiting layer in this case, is the database server.

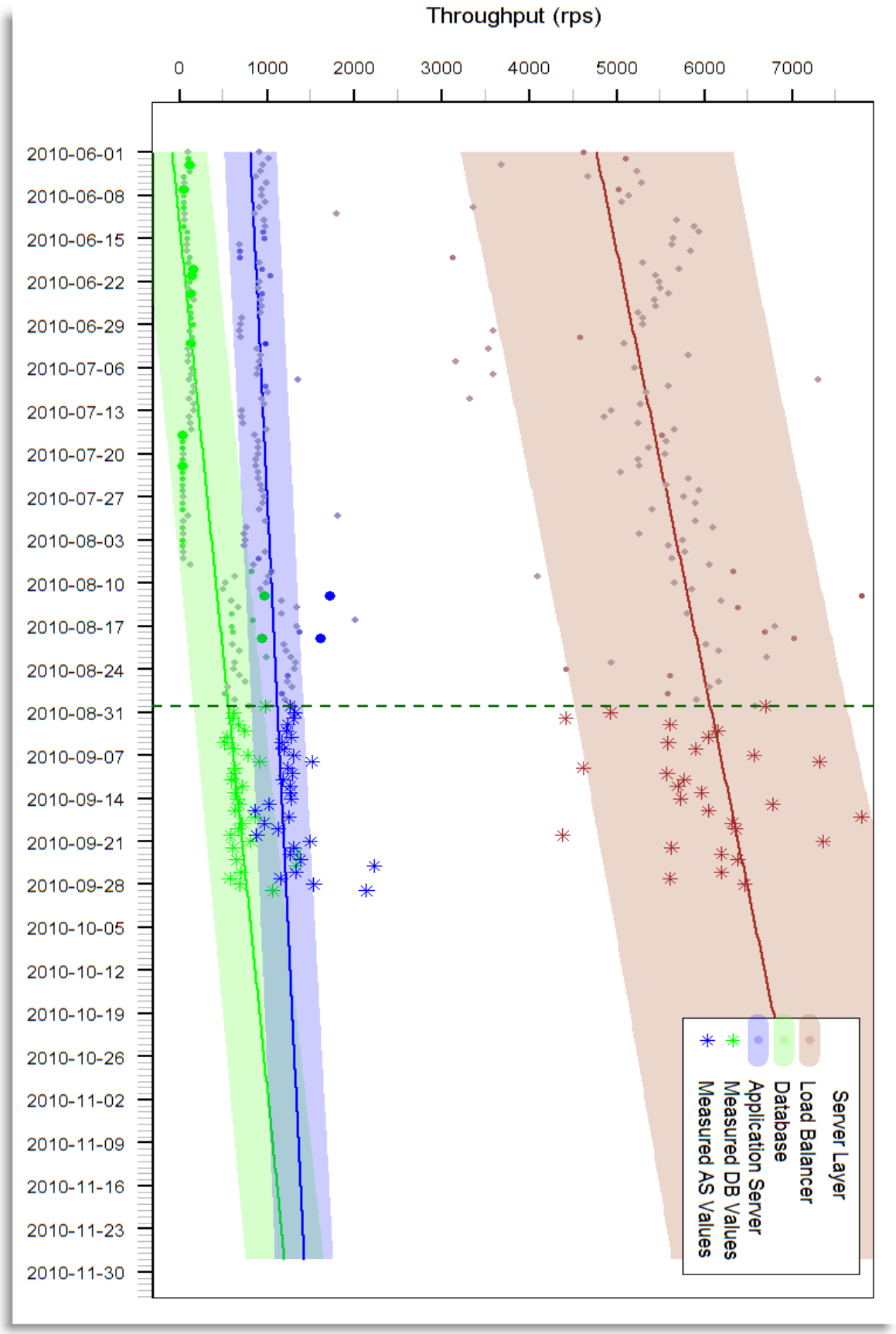

Figure 6.10: Visualisation of measured performance of different layers at the website A1 and prediction for the next 30 days

Maximum predicted throughput for each resource, i, is given for the time, t:

$$\hat{X}_i = a_i \times b_i t \tag{27}$$

where *i* is a resource for 1...n, *a* and *b* are the linear prediction coefficients of the linear regression line in Table 9.

Table 8: Summary of performance values at website A1 in a period of three months

Resource i	*Min.*	1st *Qu.*	*Median*	*Mean*	3rd *Qu.*	*Max.*
Load Balancer CPU	1790	5220	5580	5540	5910	10 500
Application Server CPU	685	895	951	1020	1190	2010
Database Server CPU	43.1	56.6	128	288	609	988

Table 9: Summary of predicted performance coefficients for equation (27) at the website A1

Resource i	*a*	*b*
Load Balancer CPU	4670	14.4
Application Server CPU	797	3.86
Database Server CPU	-102	7.67

Maximum performance of the entire server system by the time *t* is as follows:

$$\hat{X}_{max} = min(\hat{X}_i(t)) \tag{28}$$

Figure 6.11 shows the corresponding calculated performance values for network service B2. In this case, the most limiting is the application server layer. In this site there is no load balancing layer, since there is only one application server. The measured performance values are summarized in Table 10.

Predicted throughput for each resource in *i* is shown in equation *(27)*. The parameters a and b are the values of the linear prediction table according to Table 9.

Table 10: Summary of performance values at the website B2 in a period of three months

Resource	*Min.*	1st *Qu.*	*Median*	*Mean*	3rd *Qu.*	*Max.*
Application Server CPU	58.0	141	168	172	207	305
Database Server CPU	91.8	197	628	846	887	3360

Table 11: Summary of calculated performance values at the website B2

Resource i	*a*	*b*
Application Server CPU	122	1.13
Database Server CPU	267	40 921

Maximum performance of the service B2 at the time t is given by Equation *(28)* using the parameters a and b, which are shown in Table 11.

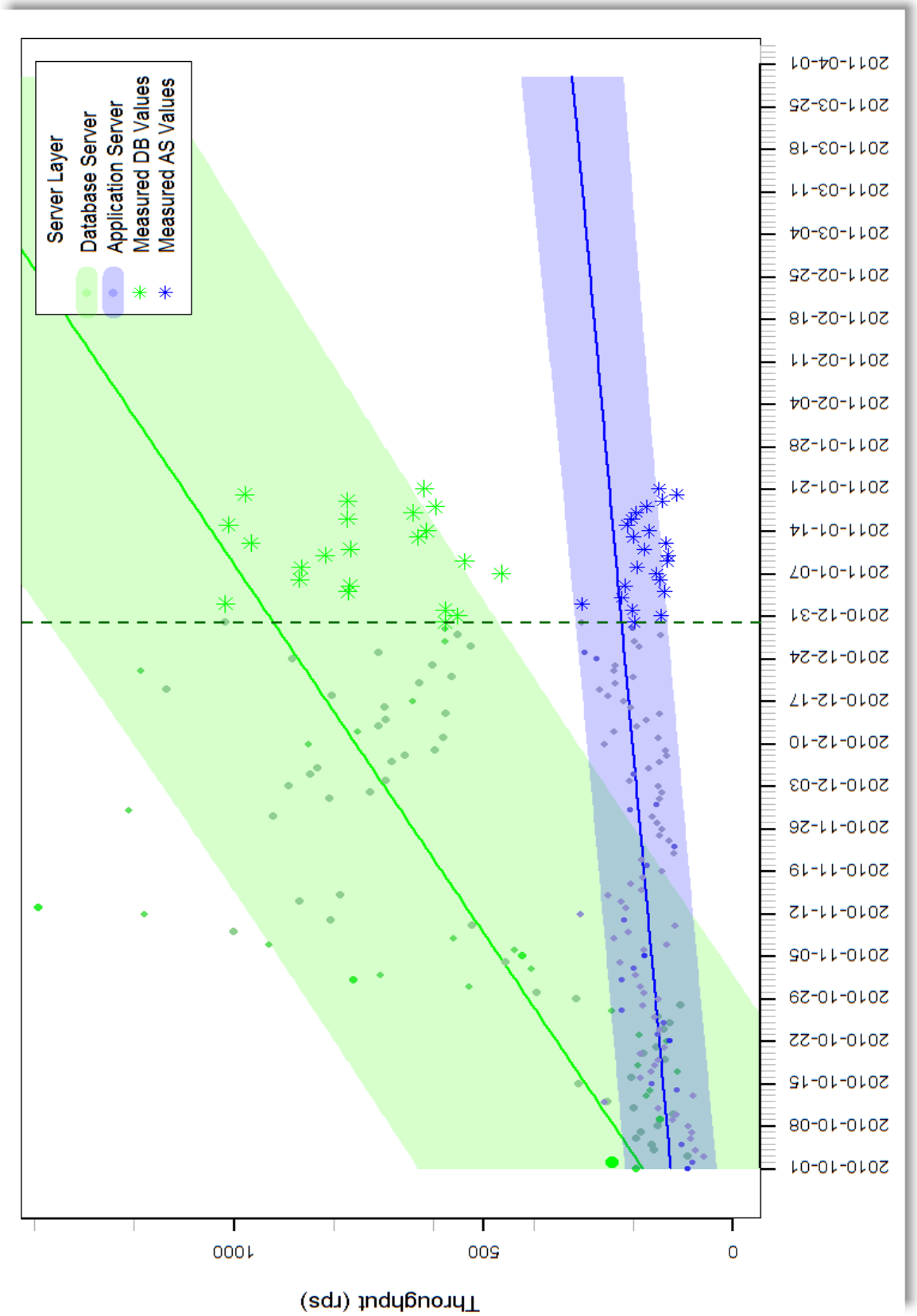

Figure 6.11: Visualisation of measured performance of different layers at the website B2 and prediction for the next 21 days

6.7 Rapid Changes in Performance and Software Aging

Sudden deterioration in performance can be interpreted in the server system as a mistake, either software or hardware or their combination. Similarly, a sudden improvement in performance is often due to increased efficiency in the new version of the software, changing configurations, the server system of internal change in status or restart of server devices or services. Slow network performance degradation can result in the software aging phenomenon. Resource consumption will continue until the available resources have all been consumed. Such an amendment is shown in Figure 2.2.

The second type of change is a progressive increase in website performance. It occurs rarely, and usually as a result of maintenance operations such as services or equipment being restarted. Figure 6.12 shows a several days long test for service A2, the result of throughput values where the virtual number of users remains constant throughout the test. As to change, the exact cause is unknown. However, it is known that the hardware or software configuration is not modified during the test. It can be assumed that the change is at least partly caused by the server or restart services or database indices' regeneration. Image of the performance change is due primarily to the fact that the response time is shortened and the virtual user sends new service requests. To begin transmission, it slowly drops until 20090414 is approximately in 400 rps level. The verse is one of the service changes, which resulted in the transmission rate suddenly rising to the level of 800 rps and then to a level of about 1200 rps.

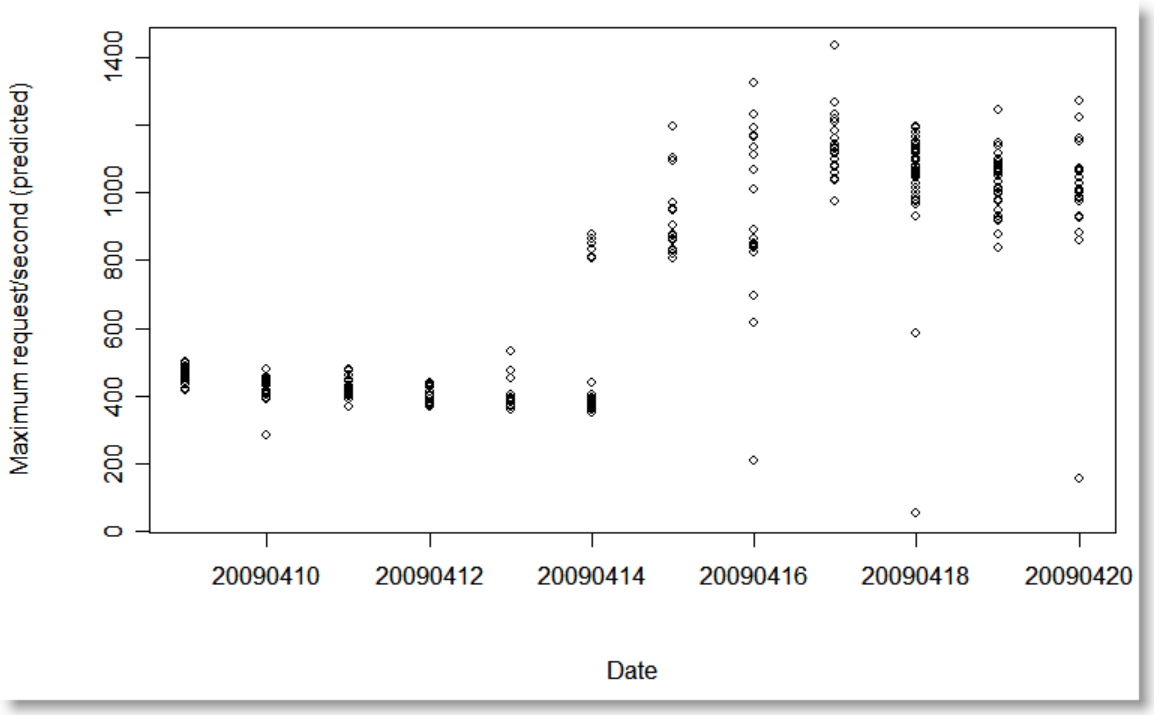

Figure 6.12: Unexpected increase in the performance of website A2

The third reason for the change is caused by long-term software maintenance and updates. A more recent version of the software may require more or even less resources than the previous one. The growing consumption of resources, aimed at the user's point of view, is for the software to provide a better level of service. Reduced consumption of resources in turn means better software internally, that is, fewer mistakes.

The tested website applications have been at the early stages of development, so the tested individual application has not been finalized in relation to resource consumption, and the phenomenon is clearly visible even for a few days during the period of lower performance.

6.8 Combination of Results

This section combines usage analysis and results of the performance prediction obtained by load test of the results, and presents them in a way easy to understand. The aim is to make the results easy to interpret; so that they can be understood by non-experts in the performance or maintenance experts of a service.

In Chapter 4, we have presented the analysis of website actual usage. It has been clarified on the basis of service, and the daily peak load causing the performance requirement. It addresses all layers of the web service. Previously, in this chapter, the analysis of current performance is shown. It shows that the service performance is determined by the different layers and particularly on the basis of the neck of the bottle of service. Combining these analyses has been clarified as a safety margin between the actual use and saturation point of the bottleneck resource.

Combining these results is shown in Figure 6.13. The green vertical line shows the measurement at a current time; the stroke on the left side is measured values, and the right side is a prediction. Red dots show the daily peak load values, which are formed on the basis of Chapter 4. The red bar shows the derived confidence interval width. Blue dots represent the system on a daily basis - the measured performance values. The blue beam is the width of the confidence interval which is derived from the measured values. Despite the fact that Figure 6.13 with the forecast extends from 90-day onwards, its accuracy is not sufficient as a maximum of 30 days. This is due to the fact that the image matching two linear extrapolation, which in themselves are relatively inaccurate.

At the right side of moment of measurement, the area between the blue and the red line is a safety margin, which exists at every moment between the predicted performance and the actual usage. The service administrator is responsible for ensuring that the service is providing a safety margin, taking into account the expected abrupt changes, which cannot be predicted on the basis of analyzed history. In some cases, the changes are predictable; some of the events come unexpectedly. Typical signs in advance of new visitors to a web store offer attractive promotions, and breaking news in media services. Both of these cause

changes in the number of visitors who do not appear in any way in the history, but their range may be more or less precisely estimated. In both cases, the number of users is generally limited in any case. Limiting factors may be marketing channel size, population of the area, language area size, etc.

If the website hardware is well balanced, the bottleneck may change to another resource on each occasion of testing. This is not relevant when assessing the adequacy of the performance of the system, or the cost of the minimum level. After wanting to make changes to the system's performance, it is also familiar with each of the server layers and the performance of each resource separately. In Figures 6.10 and 6.11 show each layer of the server performance in two different online services on the basis of processing power. The figures can be used to evaluate each layer of the server processing power’s sufficiency or excess.

Figure 6.13 contains some findings of the individual performance indices which can be interpreted as outliers. They can be caused by several reasons: inaccuracy of the measurement method, the load combinations of different surveys, the bottleneck resource is not observable among the resources, or the system's internal state is different at the initial time of tests. For these, there is no single explanation, and so these can be considered occasional uncertainty in the load-sensing system.

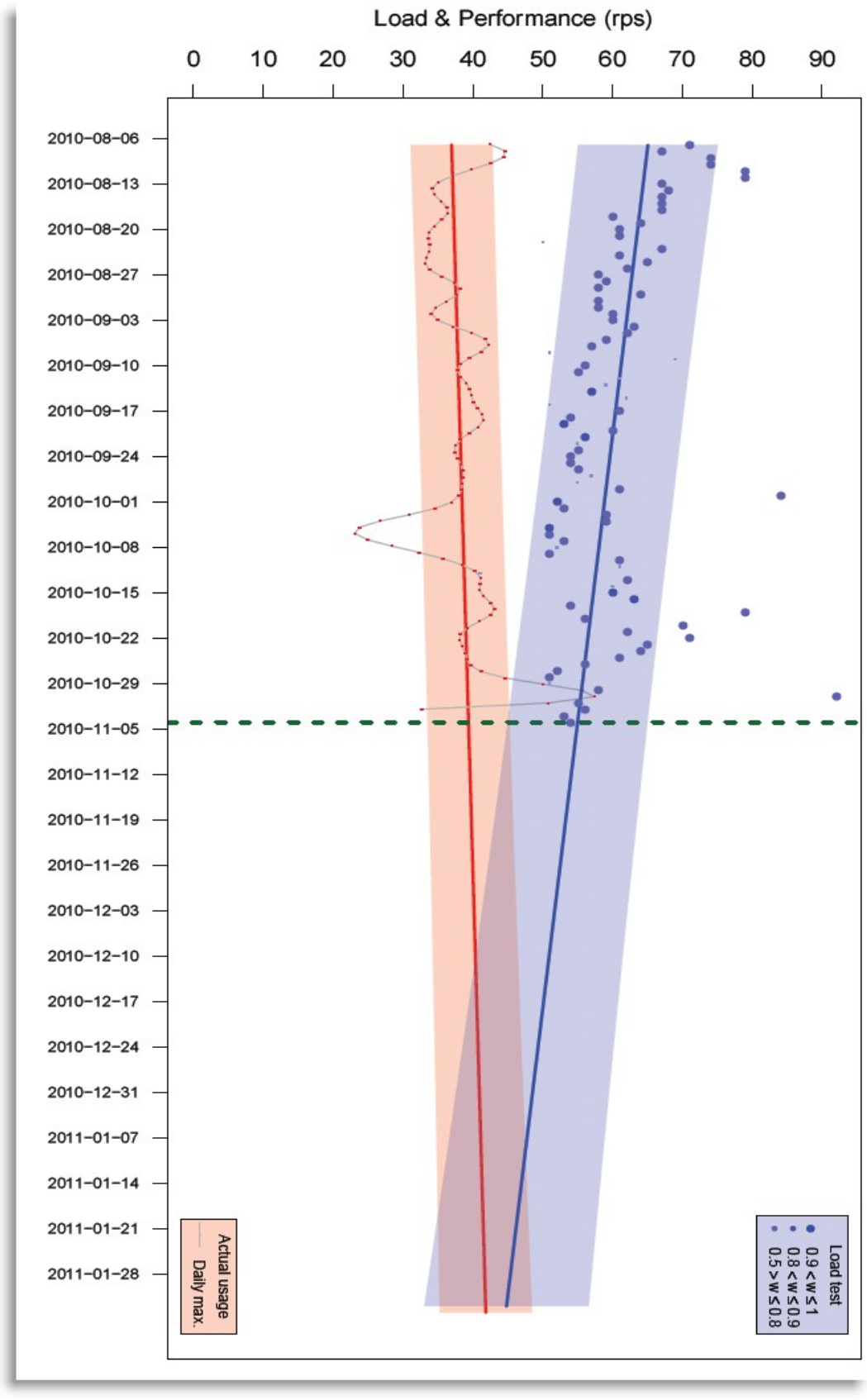

Figure 6.13: The combination of actual performance and usage of the website B.

Figure 6.10 and Figure 6.11 clearly visualize the server layers as containing too much processing power. Removal of the excess power from the server processing layers, which are not congestion, maintenance costs, and energy consumption, will be reduced without risk of interference. In addition, images with layer-specific knowledge increase the technical quality of service because the service will increase in stability.

6.9 Sensitivity Analysis

Data generated by the artificial load contains a number of factors affecting the accuracy of the outcome. This section seeks to identify those most significant in order to assess their impact on the outcome and to assess the likelihood of their occurrence, if relevant of course.

First, the meaning of precision and accuracy. The analysis aims at ensuring that the forecast accuracy is as good as possible, i.e. a method to produce the most precise estimates for each period of time in the performance of the duty cycle, which is generated by users. On the other hand, the same load of material carried out in successive runs of the performance should possibly produce high precision results to be shown as consistent with the predictions of the performance. Both the precision and accuracy are enhanced by improving the functioning of the model. In practice, the improvement of the model refers to the environment resulting from a further consideration of these factors.

Second, the results vary between the load tests, which cause the natural variation in the time series. This is shown by the increased confidence interval for the forecast. That is, on the one hand, the load test is carried out with various surveys for each test; and on the other hand, the internal state of the system is different for each time point. These variations are natural, and thus belong to a method. The large range may indicate unstable behavior in the system and the reasons leading to the need for research on a case by case basis.

Third, control of resources through the data collected requires pre-treatment. This refers to the observation of individual values (outliers) removed. These may be caused by uncontrolled external load, so the effect is filtered off. They can also be caused by a known load, in which case a single query to form an unexpectedly high, and therefore, appear in the utilization of a significant change (increase). In principle, this cannot be filtered out except for a single peak; the system does not affect the outcome. However, if the URL of the individual means the presence of a significant number, it is taken into account when forming the material of the load. In this case, the impact load occurs on the right way in the system. If any individual URL is to cause other queries compared to a very high peak load, it will turn out in the top-30 listing.

Fourth, the load and resource consumption correlate poorly or not at all. In this case, the response time is exceeded before the resource becomes saturated. This is a matter that occurs when the bottleneck resource is not found. It is evident that the bottle neck has not been any one hardware resource; instead, the application will wait for a lock semaphore or other soft resource to be released. This is not usually possible to be detected by using SNMP interface and is therefore, not possible to find in this study as shown.

Fifth, the factors resulting from the regression. The correlation between the load and resource consumption is linear up to the knee point only. The point of the knee may continue after the consumption of a linear, but the slope is higher than the former, or a change in the graph is of a higher degree. When using the load which is considerably less than the knee point, the slope of the graph is not known. On the other hand, the shape is not interesting, because the system load factor will be kept constantly under the knee-point, i.e. in the linear portion of the function.

6.10 Conclusion

In this chapter, several commercial online services have been identified through analysis of performance. The analysis is initially loaded with natural service with intermittent load queries and by monitoring the utilisation rates of the server

system resources. The results are analyzed by simple statistical methods. The results which have been found, and appear to be uniquely defined, are repeated as expected. We have found that by loading a service in a controlled manner, the resource's maximum load factor can be predicted without having the normal use of service disruption. Forecast accuracy depends on the bottleneck resource type and degree of loading. The CPU saturation prediction is fairly straightforward, but the memory usage prediction, in turn, is not due to the complexity of memory allocation algorithms. In addition, too-low a load factor does not make sense when it comes to being predictable. This is due to the fact that the measurement range varies considerably when compared to the low and high degree of loading of resources. In addition, each resource in the bottle neck, which has a low capacity and accuracy of the estimates, is not crucial.

We have also found that, although the load test is repeated several times in a short period of time in the same load, the outcome is not exactly the same performance index in all the test runs. This is due to several reasons, one of which is accompanied by a random element, which is eliminating bursts. The same set of requests does not exactly exist in different test runs. The result's range can be reduced by regression analysis. Regression analysis can be a simple linear model or a model for a better fit may be found by using a higher-order model such as Weibull Growth curve.

The experimental model seems to produce reasonably accurate and precise results for predicting the maximum performance by taking into account both the response time and the resource utilization rate.

7 Conclusions

This section presents the conclusions of the work, the answers to the research questions and the hypotheses derived from them. This section also addresses the research question and its solution.

7.1 Implications for practice

The *Predictus*-model developed in this study may be used to express the performance of a website system, especially when the performance varies for service-specific reasons. The changes can be traced from several sources. The fastest change is caused by changes in the usage, the number of users or interests that change from one page to another. The second source is the internal status of the server system, which can cause a general malfunction due to software aging. The third source is the new software versions, system software, utilities, or the application versions.

In this study, several log files are combined using a novel manner to find a prediction model, aimed at indicating the moment of the performance problem if no resolving actions are taken. The model developed in this study is applicable to predicting the forthcoming requirements for actions of a web service. Maintaining operations (hardware investments and installations, configuration changes, or software updates) can be done pro-actively and in good timing.

Performance improving techniques, such as caching, are changing the behaviour of the system within different load rates such that their functioning is difficult to model analytically or use discrete event simulation.

The method implicating the *Predictus*-model utilises several statistical methods, even concatenated. There is no aim at maximal accuracy thus far, only to prove that it is fit for use in measuring the total performance. The accuracy of the method can be intensified by increasing the accuracy at the component level. The results of the throughput are suggesting that the method gives indicative results; there is no

evidence of exact and unambiguous value for throughput. Firstly, this is caused by the inaccurate method of deriving the maximum throughput based on the measured values. Secondly, no such analytical methods exist, which could validate the measured results.

The model does not contain any protocol-specific data, so it can also be applied more widely than for the http-based systems. Here, the log analysis deals with the analysis of the http requests, so its use in connection with other protocols requires modification. Here, too, the load to perform the load test as such is not suitable for use with other protocols. JMeter, used to load on the service, is also suitable for a number of other protocols for testing, but in this study JMeter-referenced applications are not suitable for use in other protocols.

We have shown that the analysis of web server system usage can be based on ordinary http log files. The analysis can be done in a specified form in most cases, and the result is comparable day-by-day despite the nature of the web service. The actual usage of a web server is the most relevant indicator of timing and duration of the peak load. The requirement of performance on the web server system has to be based on the peak load.

The total performance of the server system consists of a unique combination of hardware, applications, configuration, and status of different applications. We have shown that the actual performance of a web server system is measurable using well-known natural load traced by web log analysis. The resource consumption can be monitored using the SNMP monitoring system. The analysis of the results can be done automatically using some simple statistical methods. We have determined that the performance curve is not linear up to the saturation point. However, we have not found a suitable higher-degree mathematical method for fitting observed performance results with the degree of utilisation within this study. The natural load should be increased up to the saturation point to figure out the asymptotic. However, at the beginning we have set the requirement not to disturb the normal usage during the load test. The utilisation up to the saturation point is the most disturbing to regular usage and leads to recovery operations in the worst case, and therefore, cannot be used. Therefore, the partial rate of load is the most recommended method for performance evaluation.

The combination of the analysis of log files and the analysis of the natural load test produces the prediction of performance for the foreseeable future. We have shown that the final result can be described in an unambiguous format so as to figure out the moment when the system does not have perform adequately to serve users with the required response times. However, the most important result is the amount of spare performance, i.e. the gap between actual usage and real performance. The user population is somewhat limited in most cases. It can be estimated based on the content, language, or even the historical knowledge of usage. In case of planned advertisement campaigns and the like, the maintainer should be able to estimate beforehand the number of expected visitors. This study has not committed itself to the unexpected change in the amount of visitors. Only the expected natural trend for visitors and performance is included.

In this method, only the http protocol is used in natural load tests and in access log analysis. This is caused by the simplicity of the logging and analysis tools. However, there are other similar protocols, like ftp and https. Other systems using connectionless protocols could be analysed as well. The connection-oriented protocols, like telnet, are rather challenging to simulate. Probably, the most troublesome feature to simulate is the think time. However, in an e-commerce environment the connection oriented protocols are rather exceptional. Another group of fairly analysable protocols is the large binary files, like multimedia streams. Yet, this group is widespread. The problem with this group is the difficulties with end-to-end response time measurement and the user experience. The former reflects differences in the system's capacity, and the latter refers to the performance of the client hardware and software, i.e. the user's computer and browser.

The exact relevancy of the shown method compared to the real web service is rather troublesome to evaluate. In the initial period, i.e. less than three months, the measured results are somewhat inconstant. Hence, the conclusions are not based on stable measurements. On the other hand, the changes since the initial period seem to stabilise or are at least explicable. In that context, the natural workload can be assumed to match the real workload. In this study, only CPU and the amount of free physical memory are monitored. The SNMP protocol makes a

great number of other resources available. Some of them are essential for an undisturbed operation, while some are devalued. The most essential question is finding the appropriate objects to monitor. When monitoring some irrelevant resources, the real bottleneck can be passed unnoticed and the results may seem unreasonable.

The study has shown that a short period of well-known natural test load is enough to define the actual performance of the server system. The length of the period is defined by the number of queries rather than the amount of time. However, all the load impulses have to be well managed, and the maximum load should be as stable as possible to increase the measurement accuracy. The study has also shown that the actual amount of incoming requests can be analysed automatically and the distribution of different queries can be found. The result of the access log analysis has been used to construct the natural work load simulation. Finally, the results have been combined automatically, and the spare capacity can be figured out. Furthermore, the moment of run out performance can be automatically defined. Lastly, the performance analysis process can be automated to achieve comparable results and avoid laborious human-oriented analysis.

The functioning of the *Predictus*-model has been tested in practice, i.e. in several websites separately. The feedback on website B has been criticized for the long response time at the early stage of the analysis series. Some improvement requirements have been revealed by the expressed model, and the measured performance was clearly becoming more effective. The extra performance removing in website type A concretizes the simultaneous change in the real and calculated performance, and hence confirms the integrity of the *Predictus*-model. Hence, the spare performance of the server system can be dissembled together with energy and other maintenance costs. In addition, the fixing of the faulty configuration, application or increase in performance, even the unexpected malfunctioning events and the loss of revenues can be avoided.

7.2 Implications for research

At the beginning of the study, three research questions and three hypotheses were identified:

> **Question 1:** *Is it possible to analyse the usage of a website and characterise it automatically and on a regular basis to identify the peak load?* The corresponding hypothesis was:
> Hypothesis 1: *The actual usage of a web server system is measurable and analysable automatically, and the result can be exploited to construct the natural load test for simulation purposes.*

Chapter 4 has shown that the service usage can be measured both quantitatively and qualitatively, and the result of measurement can be unambiguously and easily changed to communicateable information. Furthermore, a qualitative result of the measurement may be altered to such a form that can uniquely take advantage of the simulation or other analysis.

> **Question 2:** *Is it possible to measure the performance of the website system on a regular basis?* The corresponding hypothesis was:
> Hypothesis 2: *The actual performance of the server system can be defined using a known natural load with a short test-time period.*

The method presented in Section 5 is to load the web service in the same way as users of the service burden in actual use. In the same section, a method was presented that can be used to collect information on the resource consumption of the entire server system during loading and at the same time to observe the user-perceived response time. It was also discovered that the loading level can be considered to be moderate so that it does not interfere with the normal use of the service.

> **Question 3*:*** *How can the current website performance, and actual usage be compared with each other?* The corresponding hypothesis was:
> Hypothesis 3: *In combining the analysed actual usage data and the calculated total performance of the web server system, the moment when the performance of the system runs out can thus be estimated.*

7 Conclusions

The data accumulated during the artificial loading is analyzed in section 6, and the result has ascertained the ability of the service to work in actual use. When the actual usage of the website and its actual performance is converted to comparable values, the current spare performance is found. Website usage as well as variation in performance can be predicted by means of extrapolation to give a forecast of future service ability. The point at which the service is not able to effectively serve users, may be found where the curves intersect at the predicted cycle.

All the research questions have been addressed in this study, and the hypotheses have been found to be correct.

The research problem is solved. And the *Predictus*-model is applied to a data centre, it provides the opportunity to make substantial energy savings. When the performance of the server devices is adapted to a real need to match, it also eliminates overcapacity. On the other hand, it ensures that sufficient capacity is available wherever it is needed. The proposed model provides a quick and inexpensive way for performance management. Existing applications do not need to change, and hardware investments are rendered unnecessary. In addition to the removal, the over-capacity model also provides information about where the capacity is needed in order to increase the level of service that can be secured.

Results of energy savings and the management of capacity needs are commonly solved using virtualised server environments. The technology of virtualisation will probably mature in the near future. Websites located in large data centres can utilise the virtual servers most effectively. The fast performance adjustment can be arranged effectively up to the maximum total capacity of the data centres. Instead, companies maintaining a single website can benefit from the automated optimisation shown in this study.

The validity of the results must be assessed on the basis of the extent to which the meter, which is presented in the study, describes the website system measured. In order for the meter developed to be valid, it shall be described in an unbiased pattern, which is to be measured. Website usage at a certain period of time is history and it does not change over time. By increasing the size of the sample describing the use of the service, the accuracy can be improved at the same time

until finally the meter is completely unbiased. However, describing the service performance is much more difficult. It is based on the user's subjective experience of the response time, and on the other hand, response accuracy. These are not sufficient to describe the service performance. In this study, we have assumed that responses are error-free due to the performance. In addition, there are no software errors, which causes the standstill of situations that are not consuming resources.

Adequacy of resources is the primary measure of performance. When all the resources of the server system is adequate, response time is a characteristic of the system. Among other things, it depends on the physical distance between devices. Response time is only a secondary indicator, it tells us that an unrecognized resource is completely finished or will soon run out. For the SNMP interface to provide correct results in resource consumption, the method should gives a valid understanding of system performance. Different computer systems use different implementations of SNMP, and their reliability can not always be absolutely certain.

In the following, the Hevner's guidelines are used as the basis for evaluating the design research contributions of this study.

Guideline 1: Design science research must produce a viable artefact in the form of a construct, model, method, or an instantiation.

- The *Predictus*-model and any method that has been built on the basis of the model, form artefact according to Hevner's first guideline.
- IT artefacts are rarely fully prepared for information systems; in most cases, they are innovations. So even in this case, the *Predictus*-model is a guideline for the development of a methodology. The final accuracy, precision, and usefulness of the performance prediction is contingent upon the reliability of the means used.

Guideline 2: The objective of design science research is to develop technology-based solutions to important and relevant business problems.

- With method, the end result is awareness of the 'performance reserve' size, which is an important business information. It can be used to draw the

necessary conclusions over performance increase or decrease, or the timing of the necessary activities.

Guideline 3: The utility, quality, and efficacy of a design artefact must be rigorously demonstrated via well-executed evaluation methods.

- In Figure 1.3 the *Predictus*-model is the result of a long iterative process. The development is guided by the awareness that the result should be beneficial, both economically and technically.
- A more detailed discussion is found at each stage of the method in the sensitivity analysis (sections 4.5, 5.4, and 6.9). They deal with the model and the method's accuracy-related issues.

Guideline 4: Effective design science research must provide clear and verifiable contributions in the areas of design artefact, design foundations, and/or design methodologies.

- As set out above, the artefact gives a clear answer to the research problem. As seen in further studies (section 7.3), the result's accuracy can be improved by improving the method.
- The most important contribution of this study, however, is the result emanating from the *Predictus*-model development. It can clearly and regularly provide information on the level of spare performance at each time of measurement.

Guideline 5: Design science research relies upon the application of rigorous methods in both the construction and evaluation of the design artefact.

- Data collection and calculation methods used in this thesis, are commonly used. According to the results, their accuracy is meant in many ways to attract attention. Changes in system performance, caused by the collection of data, are discussed in section 4.1. The accuracy of results provided by SNMP interface has been thoroughly discussed in section 5.3. Performance relevancy and accuracy have been assessed in section 6.5 using the R-value, and in section 6.8 in terms of spare performance.

Guideline 6: The search for an effective artefact requires utilizing every available means to reach the desired ends while satisfying laws in the problem environment.

- The *Predictus*-model development process is described briefly in the introductory chapter; it took place during a period of several years of trial-and-error. The design process has not been particularly systematic, little things have had to be resolved on a case-by-case basis.

Guideline 7: Design science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

- The method produces a concrete performance prediction which can be communicated on a regular basis to management and technically-oriented staff so that the necessary conclusions and actions can be drawn.

7.3 Limitations of the research and suggestions for further studies

One of the possible limitations of this research is that the adoption study has been conducted for a single protocol. No statistical generalisation of the results can be provided in a single case research and the representativeness of the results in other environments may be questionable. However, the concentration on one protocol has brought with it the benefit of allowing for an in-depth study.

The work utilizes two independent combinations of linear extrapolation. When two low-precision projections are combined, the result may not be particularly accurate. The result is indicative, in many cases sufficient, but far-reaching conclusions about the final outcome can not be drawn. The study has not taken into account in any way the inaccuracy which is immediately followed by a change of system performance or website usage.

The study is focused on dealing with reasonably short responses, such as a normal web page in general. When handling a variety of video or audio streams, or other similar continuous data streams, the method is not apparently suitable for performance evaluation. In such a case, the response time is poorly suited for describing user experience.

The method may be further developed in several different directions. One obvious target for development is a protocol independency. The other is the quality of experienced by the user to be described in an alternative way rather than the response time. In addition, the accuracy of analysis can be improved by introducing a more sophisticated analysis rather than the linear extrapolation, used throughout this study.

References

APM Group Ltd. (2011). ITIL official website. Retrieved from http://www.itil-officialsite.com/

Aalto-www. (2010, December). Uutiset: Aalto-yliopistossa kehitetään energiatehokkaita palvelinkeskuksia Suomeen - Aalto-yliopisto. Retrieved December 8, 2010, from http://www.aalto.fi/fi/current/news/view/2010-12-02/

Aken, J. (2004). Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *Journal of management studies*, *41*(2), 219-246.

Andreolini, A., Cardellini, V., & Colajanni, M. (2002). Benchmarking Models and Tools for Distributed Web-Server Systems. *Performance 2002*, 208-235.

Andreolini, M., & Casolari, S. (2006). Load prediction models in web-based systems. *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools - valuetools '06* (p. 27). New York, New York, USA: ACM Press. doi:10.1145/1190095.1190129

Apache Software Foundation. (n.d.). Welcome! - The Apache Software Foundation. Retrieved August 18, 2010, from http://www.apache.org/

Ardaiz, O., Freitag, F., & Navarro, L. (2001). Estimating the service time of web clients using server logs. *ACM SIGCOMM Computer Communication Review*, *31*(2 supplement), 108. doi:10.1145/844193.844202

Arlitt, Martin, Krishnamurthy, D., & Rolia, J. (2001). Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, *1*(1), 44-69. doi:10.1145/383034.383036

References

Avritzer, A., Farel, R., Futamura, K., Hosseini-Nasab, M., Karasaridis, A., Mainkar, V., Meier-Hellstern, K., et al. (2002). Performance Analysis In the Age of the Internet: A New Paradigm for a New Era.

Bacigalupo, D. A., Jarvis, S. A., & Nudd, G. R. (2004). An investigation into the application of different performance prediction techniques to e-commerce applications. *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* (pp. 248-255). Santa Fe, NM, USA: IEEE. doi:10.1109/IPDPS.2004.1303306

Bagchi, S., Hung, E., Iyengar, A., Vogl, N., & Wadia, N. (2006). Capacity planning tools for web and grid environments. *Proceedings of the 1st international conference on Performance evaluation methodolgies and tools - valuetools '06* (p. 25). New York, New York, USA: ACM Press. doi:10.1145/1190095.1190127

Balaton, Z., Kacsuk, P., Podhorszki, N., & Vajda, F. (2000). Comparison of representative grid monitoring tools. *Reports of the Laboratory of Parallel and Distributed Systems (SZTAKI), LPDS-2/2000*. Citeseer.

Barford, P., & Crovella, M. (1998). Generating representative Web workloads for network and server performance evaluation. *ACM SIGMETRICS Performance Evaluation Review*, *26*(1), 151-160. doi:10.1145/277858.277897

Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M., & Yimwadsana, T. (2005). Predictability of Web-Server Traffic Congestion. *10th International Workshop on Web Content Caching and Distribution (WCW'05)* (pp. 97-103). Sophia Antipolis, France: IEEE. doi:10.1109/WCW.2005.17

References

Benedetto, S., Correia, L. M., Luise, M., Bogucka, H., Palicot, J., & Moy, C. (2012). Green communications (pp. 169-179). Springer Milan. doi:10.1007/978-88-470-1983-6_9

Berners-Lee, T., Caillan, R., Luotonen, A., Nielesen, H. F., & Secret, A. (1994). The World-Wide Web. *Communications of the ACM*, *37*, 77-82.

Bhulai, S., Sivasubramanian, S., van der Mei, R., & van Steen, M. (2007). Modeling and Predicting End-to-End Response Times in Multi-tier Internet Applications. In T. D. L. Mason & J. Yan (Eds.), *ITC20'07 Proceedings of the 20th international teletraffic conference on Managing traffic performance in converged networks* (pp. 519-532). Springer-Verlag Berlin Heidelberg.

Bi, Y., Zhao, J., & Zhang, D. (2004). Power load forecasting algorithm based on wavelet packet analysis. *Power System Technology, 2004. PowerCon 2004. 2004 International Conference on* (Vol. 1, pp. 987–990). IEEE.

Bloom, B. (1956). The Classification of Educational Goals by a Committee of College and University Examiners. *Taxonomy of Educational Objectives, Handbook I*, *1*. New York: Longmans.

Booth, G. M. (1976). Distributed information systems. *Proceedings of the June 7-10, 1976, national computer conference and exposition* (pp. 789–794). ACM.

Bouch, A., Kuchinsky, A., & Bhatti, N. (2000). Quality is in the eye of the beholder. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00* (pp. 297-304). New York, New York, USA: ACM Press. doi:10.1145/332040.332447

Box, G. E. P., & Jenkins, G. M. (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day Inc., 500 Sansome Street, San Francisco, California, USA.

References

Buckley, J., & Exton, C. (2003). Bloom's taxonomy: a framework for assessing programmers' knowledge of software systems. *MHS2003. Proceedings of 2003 International Symposium on Micromechatronics and Human Science (IEEE Cat. No.03TH8717)* (pp. 165-174). Portland, OR, USA: IEEE Comput. Soc. doi:10.1109/WPC.2003.1199200

Calzarossa, M., Italiani, M., & Serazzi, G. (1986). A workload model representative of static and dynamic characteristics. *Acta Informatica*, *23*(3), pp. 255-266. doi:10.1007/BF00289113

Cameron, C. A., & Windmeijer, F. A. G. (1997). An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics*, *77*(2), pp. 329-342. doi:10.1016/S0304-4076(96)01818-0

Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991). The information visualizer, an information workspace. *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology - CHI '91* (pp. 181-186). New York, New York, USA: ACM Press. doi:10.1145/108844.108874

Case, J., Fedor, M., Schoffstall, M., & Davin, J. (1990). *Simple Network Management Protocol (SNMP)*. Retrieved from http://www.ietf.org/rfc/rfc1157.txt

Castelli, V., Harper, R. E., Heidelberger, P., Hunter, S. W., Trivedi, K. S., Vaidyanathan, K., & Zeggert, W. P. (2001). Proactive management of software aging. *IBM Journal of Research and Development*, *45*(2), pp. 311-332. doi:10.1147/rd.452.0311

Cherkasova, L., Fu, Y., & Tang, W. (2002). Measuring end-to-end internet service performance: Response time, caching efficiency and QoS. Retrieved from http://www.hpl.hp.com/techreports/2002/HPL-2002-148.pdf

References

Ciemiewicz, D. M. (2001). What Do You Mean?-Revisiting Statistics for Web Response Time Measurements. *CMG-CONFERENCE-* (Vol. 1, pp. 385–396). Computer Measurement Group; 1997. Retrieved from http://www.cmg.org/proceedings/2001/1303.pdf

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A Seasonal-Trend Decomposition Procedure Based on Loess. *Journal of Official Statistics*, *6*, pp. 3-73.

Coffman, K. G., & Odlyzko, A. M. (2001). Internet growth: Is there a "Moore's Law" for data traffic? *Handbook of massive data sets* (Vol. 142, pp. 47-93). Kluwer. Retrieved from http://papers.ssrn.com/sol3/papers.cfm?abstract_id=236108

Commission of the European Communities. (2008). Addressing the challenge of energy efficiency through Information and Communication Technologies. *Communication from the Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions*. Retrieved June 10, 2008, from http://www.scribd.com/doc/6407079/Addressing-the-Challenge-of-Energy-Efficiency-Through-Information-and-Communication-Technologies-COM2008-241EN

Crovella, M. E., & Bestavros, A. (1997). Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Transactions on Networking* (Vol. 5, pp. 835-846). doi:10.1109/90.650143

Datacenter Dynamics. (2011). Forecasting Energy Demand. Retrieved from http://www.datacenterdynamics.com/research/energy-demand-2011-12

DeveloperSide.NET. (n.d.). Apache Performance Tuning. Retrieved August 18, 2010, from http://www.devside.net/articles/apache-performance-tuning

References

Dill, S., Kumar, R., Mccurley, K. S., Rajagopalan, S., Sivakumar, D., & Tomkins, A. (2002). Self-similarity in the web. *ACM Transactions on Internet Technology*, *2*(3), 205-223. doi:10.1145/572326.572328

Dilley, J., Friedrich, R., Jin, T., & Rolia, J. (1998). Web server performance measurement and modeling techniques. *Performance Evaluation*, *33*(1), 5-26. doi:10.1016/S0166-5316(98)00008-X

Draheim, D., Grundy, J., Hosking, J., Lutteroth, C., & Weber, G. (2006). Realistic load testing of Web applications. *Conference on Software Maintenance and Reengineering (CSMR'06)* (p. 11 pp.-70). Bari, Italy: IEEE. doi:10.1109/CSMR.2006.43

Dumke, R., Rautenstrauch, C., Schmietendorf, A., & Scholz, A. (2001). *Performance engineering: state of the art and current trends*. Heidelberg: Springer-Verlag.

Eusgeld, I., Happe, J., Limbourg, P., Rohr, M., & Salfner, F. (2008). Performability. In I. Eusgeld, F. Freiling, & R. Reussner (Eds.), *Dependability Metrics* (Vol. 4909, pp. 245-254). Springer Berlin / Heidelberg.

Ferrari, D. (1972). Workload charaterization and Selection in Computer Performance Measurement. *Computer*, *5*(4), 18-24. doi:10.1109/C-M.1972.216939

Ferrari, D. (1983). *Measurement and tuning of computer systems*. Englewood Cliffs NJ: Prentice-Hall.

Ferrari, G., Ezhilchelvan, P., & Mitrani, I. (2006). Performance Modeling and Evaluation of E-Business Systems. *39th Annual Simulation Symposium (ANSS'06)* (pp. 135-142). IEEE. doi:10.1109/ANSS.2006.36

References

Fit4Green Consortium. (2010). Fit4Green | Energy aware ICT optimization policies. Retrieved December 14, 2010, from http://www.fit4green.eu/

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

Galletta, D. (2002). Web site delays: How tolerant are users? *Journal of the Association of Information Systems*, (5(1)), 1-28. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.1775&rep=rep1&type=pdf

Garg, S., van Moorsel, A., Vaidyanathan, K., & Trivedi, K. S. (1998). A methodology for detection and estimation of software aging. *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)* (pp. 283-292). IEEE Comput. Soc. doi:10.1109/ISSRE.1998.730892

Goel, A. (2004). *Advances in Distributed Systems*. University of Toronto.

Google Inc. (2010). Google Analytics | Official Website. Retrieved December 3, 2010, from http://www.google.com/analytics/

Gray, J., & Siewiorek, D. P. (1991). High-availability computer systems. *Computer*, *24*(9), 39–48. IEEE. doi:10.1109/2.84898

GreenICT.com.au. (2008). GreenICT.com.au. Retrieved from http://greenict.com.au/

GreenICT.org.uk. (n.d.). GreenICT.org.uk. Retrieved from http://greenict.org.uk/

References

Grottke, M., Li, L., Vaidyanathan, K., & Trivedi, K. S. (2006). Analysis of Software Aging in a Web Server. *IEEE Transactions on Reliability*, *55*(3), 411-420. doi:10.1109/TR.2006.879609

Hadharan, R., Ehrlich, W. K., Cura, D., & Reeser, P. K. (2000). End to End Performance Modeling of Web Sewer Architectures. *ACM SIGMETRICS Performance Evaluation Review*, *28*(2), 57-63. doi:10.1145/362883.581258

Haring, G. (1983). On stochastic models of interactive workloads. *Proceedings of the 9th International Symposium on Computer Performance Modelling, Measurement and Evaluation* (Vol. 1983, pp. 133–152). Amsterdam;New York ▪ ;New York N.Y.: North-Holland Publishing Co.

Hellerstein, J. L., Zhang, F., & Shahabuddin, P. (1998). Characterizing normal operation of a web server: Application to workload forecasting and problem detection. *CMG-CONFERENCE*- (Vol. 1, pp. 150–160). COMPSCER MEASUREMENT GROUP INC.

Hevner, A., March, S., & Park, J. (2004). Design science in information systems research. *Mis Quarterly*. Retrieved from http://dl.acm.org/citation.cfm?id=2017217

Hu, J., Deng, J., & Wu, J. (2011). A Green Private Cloud Architecture with global collaboration. *Telecommunication Systems*. doi:10.1007/s11235-011-9639-5

Huang, Y. (1995). Software Rejuvenation: Analysis, Module and Applications. *FTCS '95 Proceedings of the Twenty-Fifth International Symposium on Fault-Tolerant Computing*.

IDC. (2007). Virtualization and Multicore Innovations Disrupting the Worldwide Server Market, According to IDC. Retrieved February 15, 2012, from

http://virtualization.info/en/news/2007/03/idc-reports-virtualization-and.html

ISO/IEC. (2001, June). *ISO9126-1. Software engineering - Product quality - Part 2: External Metrics*.

Imai, K., King, G., & Lau, O. (2006). Zelig: Everyone's Statistical Software. *R package version*, 2–7. Citeseer.

Imai, K., King, G., & Lau, O. (2008). Toward A Common Framework for Statistical Analysis and Development, *17*(4), 892-913.

Iosup, A., Epema, D. H. J., Franke, C., Papaspyrou, A., Schley, L., Song, B., & Yahyapour, R. (2007). On Grid Performance Evaluation Using Synthetic Workloads. In E. Frachtenberg & U. U. Schwiegelshohn (Eds.), *SSPP'06 Proceedings of the 12th international conference on Job scheduling strategies for parallel processing* (Vol. 4376, pp. 232-255). Springer-Verlag Berlin Heidelberg.

JMeter contributors. (n.d.). JMeter - Apache JMeter. Retrieved October 24, 2010, from http://jakarta.apache.org/jmeter/

Jewell, D. (2008). Performance Engineering and Management Method — A Holistic Approach to Performance Engineering. *Performance Modeling and Engineering*, 29–55. Boston, MA: Springer.

Jian, R. (1991). The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling. John Wiley & Sons, Inc., New York, USA.

Jiang, S. (2012). Green Networking Strategies Versus Networking Modes. *Future Wireless and Optical Networks*.

References

Järvinen, P. H. (2004). *On Research Methods*. Tampere: Opinpaja.

Kaplan, J., Forrest, W., & Kindler, N. (2008). Revolutionizing data center energy efficiency. *McKinsey & Company, Tech. Rep*.

Koomey, J. G. (2007). Estimating total power consumption by servers in the US and the world. Oakland, CA: Analytics Press.

Koomey, J. G. (2011). Growth in Data Center Electricity Use 2005 to 2010. Oakland, CA. Retrieved from http://www.analyticspress.com/datacenters.html

Kotsis, G. (2004). Performance Management in Dynamic Computing Environments. *Performance Tools and Applications to Networked Systems*, 254–264.

Koziolek, H. (2008). Introduction to Performance Metrics. In I. Eusgeld, F. Freiling, & R. Reussner (Eds.), *Dependability Metrics* (Vol. 4909, pp. 199-203). Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-68947-8_17

Koziolek, H., & Happe, J. (2008). Performance Metrics for Specific Domains. In I. Eusgeld, F. Freiling, & R. Reussner (Eds.), *Dependability Metrics* (Vol. 4909, pp. 233-240). Springer Berlin / Heidelberg.

Kuhmann, W. (1989). Experimental investigation of stress inducing properties of system response times. *Ergonomics, 32*, 271-280.

Kuhmann, W., Boucsein, W., Schaefer, F., & Alexander, J. (1987). Experimental investigation of psychophysiological stress-reactions induced by different system response times in human-computer interaction. *Ergonomics*, *30*, 933-943.

References

Lightner, N. J., & Zeng, L. (2009). What is still wrong with the World-Wide Web? An update after a decade. *Journal of Intelligent Manufacturing*, *22*(1), 3-15. doi:10.1007/s10845-009-0275-9

Lilja, D. J. (2000). *Measuring computer performance: a practitioner's guide*. New York, NY: Cambridge Univ Press. Retrieved from http://books.google.com/books?hl=en&lr=&id=jb68T-OuIC4C&oi=fnd&pg=PP1&dq=Measuring+computer+performance:+a+practitioner%27s+guide&ots=XTORdPwObq&sig=21hb17-M0lD1KrhiShvmCdXhLQg

Lin, W., Liu, Z., Xia, C. H., & Zhang, L. (2005). Optimal capacity allocation for Web systems with end-to-end delay guarantees. *Performance Evaluation*, *62*(1-4), 400-416. doi:10.1016/j.peva.2005.07.021

Lindgaard, G., & Dudek, C. (2002). What is this evasive beast we call user satisfaction? *Interacting with Computers*, 492-452.

Little, J. D. C. (1961). A Proof for the Queueing Formula: L=Lamdba W. *Operations Research*, *9*, 383-387.

Loosley, C. (2005). *When Is Your Web Site Fast Enough?* Retrieved from http://www.ecommercetimes.com/story/46627.html

Lu, S., Yang, X., & Zhao, X. (2004). The application of modeling and prediction with MRA wavelet network. *Journal of Marine Science and Application*, *3*(1), 20-23. doi:10.1007/BF02918641

Mahanti, A., Williamson, C., & Wu, L. (2009). Workload Characterization of a Large Systems Conference Web Server. *2009 Seventh Annual Communication Networks and Services Research Conference* (pp. 55-64). Moncton, BC, Canada: IEEE. doi:10.1109/CNSR.2009.19

References

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems*, *15*(4), 251–266. Elsevier.

Martin, G., & Corl, K. (1986). System response time effects on user productivity. *Behavior and Information Technology*, *5*, 3-13.

Menascé, D. (2000). *Scaling for e-business: technologies, models, performance, and capacity planning*. Upper Saddle River NJ: Prentice Hall PTR.

Menascé, D. A., & Almeida, V. A. F. (2002). *Capacity Planning for Web Services*. Prentice Hall, Inc., New Jersey, USA.

Menascé, D. A., Almeida, V. A. F., Fonseca, R., & Mendes, M. A. (1999). A methodology for workload characterization of E-commerce sites (pp. 119-128). Denver, Colorado, United States. doi:10.1145/336992.337024

Menascé, D. A., Almeida, V. A. F., Riedi, R., Ribeiro, F., Fonseca, R., & Meira, W. (2003). A hierarchical and multiscale approach to analyze E-business workloads. *Performance Evaluation*, *54*(1), 33-57. doi:10.1016/S0166-5316(02)00228-6

Microsoft Corporation. (n.d.). The Official Microsoft IIS Site. Retrieved August 18, 2010, from http://www.iis.net/

Mielke, A. (2006). Elements for response-time statistics in ERP transaction systems. *Performance Evaluation*, *63*(7), 635-653. doi:10.1016/j.peva.2005.05.006

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)* (Vol. 33, p. 267). New York, New York, USA: ACM Press. doi:10.1145/1476589.1476628

References

Mindcraft. (2010). Mindcraft - WebStone Benchmark Information. Retrieved December 3, 2010, from http://www.mindcraft.com/webstone/

Miniwatts Marketing Group. (2012). Internet Users in the World. *Internet World Stats*. Retrieved from http://www.internetworldstats.com/stats.htm

Mullender, S. (1993). Distributed systems. University of Edinburgh. Retrieved from http://doc.utwente.nl/64628

Net-SNMP Development Team. (2007). Net-SNMP. Retrieved January 2, 2011, from http://www.net-snmp.org/

OMG (Object Management Group). (2011). BPMN Specification. Retrieved from http://www.bpmn.org/

Overbaugh, R. C. (n.d.). Bloom's Taxonomy. Retrieved November 25, 2010, from http://www.odu.edu/educ/roverbau/Bloom/blooms_taxonomy.htm

Papagiannaki, K., Taft, N., Zhang, Z.-L., & Diot, C. (2005). Long-term forecasting of Internet backbone traffic: observations and initial models. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)* (Vol. 2, pp. 1178-1188). IEEE. doi:10.1109/INFCOM.2003.1208954

Podelko, A. (2007, January). Performance Testing and Performance Engineering. Retrieved January 13, 2011, from http://www.testingreflections.com/node/view/4816

Podelko, A. (2009, September). Performance Testing Innovations. Retrieved January 13, 2011, from http://www.testingreflections.com/node/view/8285

References

Raghavan, S., Vasukiammaiyar, D., & Haring, G. (1993). *Generative models for network load in a single server environment*. University of Maryland, College Park.

Ruffo, G., Schifanella, R., Sereno, M., & Politi, R. (2004). WALTy: a user behavior tailored tool for evaluating web application performance. *Third IEEE International Symposium on Network Computing and Applications, 2004. (NCA 2004). Proceedings.* (pp. 77-86). IEEE. doi:10.1109/NCA.2004.1347765

SPEC. (2010). SPEC - Standard Performance Evaluation Corporation. Retrieved January 3, 2011, from http://www.spec.org/

Sang, A., & Li, S. (2000). A predictability analysis of network traffic. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)* (Vol. 1, pp. 342-351). IEEE. doi:10.1109/INFCOM.2000.832204

Schäppi, B., Bellosa, F., Przywara, B., Bogner, T., & Weeren, S. (2007). *Energy efficient servers in Europe Energy consumption, saving potentials, market barriers and measures*. Retrieved from http://www.efficient-server.eu/fileadmin/docs/reports/E-Server_PartI_SavingPotentials_and_Scenarios_28112007.pdf

Selhofer, H., Lilischkis, S., Woerndl, M., Alkas, H., & O'Donnell, P. (eds.). (2008). *The European e-Business Report 2008*. Retrieved from http://www.ebusiness-watch.org/key_reports/documents/EBR08.pdf

Selvidge, P. R., Chaparro, B. S., & Bender, G. T. (2002). The world wide wait: effects of delays on user performance. *International Journal of Industrial Ergonomics, 29*, 15-20.

References

Silva, L. H. M. (2006). Software Aging and Rejuvenation in a SOAP-based Server. *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)* (Vol. 2006, pp. 56-65). Cambridge, MA, USA: IEEE. doi:10.1109/NCA.2006.51

Singleton, P. (2002, July). Performance Modelling — What, Why, When and How. *BT Technology Journal*, *2002*(20). doi:10.1023/A:1020860029447

Soininen, J., & Jaakkola, H. (2012). Knowledge Mining of the Web Services Usage. *Proceedings of The 22nd European Japanese Conference on Information Modelling and Knowledge Bases (EJC 2012), June 4-8, 2012, Final version in Information Modelling and Knowledge Bases XXIV, IOS Press, Amsterdam. Accepted for publication.* Prague, Czech Republic: IOS Press.

Splaine, S., & Jaskiel, S. R. (2001). *The Web Testing Handbook*. STQE Publishing. doi:10.1109/TPC.2011.2182569

Sugiyama, Y. (2012). Green ICT toward Low Carbon Society. *Design for Innovative Value Towards a Sustainable Society Proceedings of EcoDesign 2011: 7th International Symposium on Environmentally Conscious Design and Inverse Manufacturing*. doi:10.1007/978-94-007-3010-6_149

Sullivan, M., & Chillarege, R. (1991). Software defects and their impact on system availability-a study of field failures in operating systems. *[1991] Digest of Papers. Fault-Tolerant Computing: The Twenty-First International Symposium* (pp. 2-9). IEEE Comput. Soc. Press. doi:10.1109/FTCS.1991.146625

TPC. (2010). TPC - Homepage. Retrieved January 3, 2011, from http://www.tpc.org/

References

Team, R. D. C. (2011). *R: A Language and Environment for Statistical Computing*. Vienna, Austria. Retrieved from http://www.r-project.org

Thalheim, B., & Tropmann, M. (2011). Performance Forecasting for Performance Critical Huge Databases. *Proceeding of the 2011 conference on Information Modelling and Knowledge Bases XXII* (pp. 206–225). IOS Press. Retrieved from http://dl.acm.org/citation.cfm?id=1972767

Tierney, B., Crowley, B., Gunter, D., Lee, J., & Thompson, M. (2001). A monitoring sensor management system for grid environments. *Cluster Computing*, *4*(1), 19–28. Springer. doi:10.1023/A:1011408108941

Tierney, B., Johnston, W., Crowley, B., Hoo, G., Brooks, C., & Gunter, D. (1998). The NetLogger methodology for high performance distributed systems performance analysis. *High Performance Distributed Computing, 1998. Proceedings. The Seventh International Symposium on* (pp. 260–267). IEEE.

Titchkosky, L., Arlitt, M., & Williamson, C. (2003). A performance comparison of dynamic web technologies. *ACM SIGMETRICS Performance Evaluation Review*, *31*(3), 2–11. ACM.

Tsai, C.-H., Shin, K. G., Reumann, J., & Singhal, S. (2007). Online web cluster capacity estimation and its application to energy conservation. *Parallel and Distributed Systems, IEEE Transactions on*, *18*(7), 932–945. IEEE. doi:10.1109/TPDS.2007.1028

VMware, I. (n.d.). Performance Best Practices for VMware vSphere™ 4.1. Retrieved November 16, 2011, from http://www.vmware.com/pdf/Perf_Best_Practices_vSphere4.1.pdf

References

Van Steen, M. (2003). Distributed Systems Principles and Paradigms. *Network*, *1*, 2. Prentice-Hall. Retrieved from http://www.distributed-systems.net/courses/ds-slides/notes.01.pdf

Vercauteren, T., Aggarwal, P., Wang, X., & Li, T. H. (2007). Hierarchical forecasting of web server workload using sequential Monte Carlo training. *Signal Processing, IEEE Transactions on*, *55*(4), 1286–1297. IEEE.

Walker, R. (2006). Examining Load Average | Linux Journal. Retrieved December 7, 2010, from http://www.linuxjournal.com/article/9001

Weiss, S., Boggs, G., Lehto, M., Shodja, S., & Martin, D. (1982). *Computer system response time and psycho-physiological stress II* (pp. 698-702).

Yang, R., & Theys, M. D. (2005). RMF: Resource monitoring framework for integrating active and passive monitoring tools in Grid environments. *Journal of Parallel and Distributed Computing*, *65*(11), 1419–1428. Elsevier.

Zhang, Z., & Fan, W. (2008). Web server load balancing: A queueing analysis. *European Journal of Operational Research*, *186*(2), 681–693. Elsevier.

Glossary of Terms and Abbreviations

Accuracy

The absolute difference between a reference value and the corresponding reference value

ANalysis Of VAriance (ANOVA)

a general statistical technique used to separate the total variation observed in a set of measurements into the variation due to measurement error within each alternative and the variation across alternatives

Analytic model

A modeling technique that uses mathematical expressions to represent relationships between modeled system components

Arithmetic mean

The sum of all values divided by the number of values

ASymmetric MultiProcessor (ASMP)

A synonym for “loosely-coupled multiprocessor”

Availability

a metric used to represent the percentage of time a system is available during an observation period

Bandwidth

the maximum possible throughput of a resource

Benchmark

A well-defined, repeatable workload that can be executed on various systems in order to compare performance

Bottleneck

A resource that saturates first as the workload intensity increases. it is the resource with the highest service demand

Cache

A small fast memory holding recently-accessed data, designed to speed up subsequent accesses to the same data. A local data structure holding a copy of remote data

Capacity

Capacity is about some activity over time, e.g. bytes moved over a period, from which utilisation can be derived. Capacity is e.g. the number of cars per hour a motorway can handle.

Capacity analysis

Evaluation of a factory, production process or line, or machine, to determine its maximum output rate.

Capacity management

The process of ensuring the current capacty is adequate and used in the most effective way

Capacity planning

A process of predicting when future load levels will saturate the system and of determining the most cost-effective way of delaying system saturation as much as possible

Central Processing Unit (CPU)

The arithmetic, logic, and control unit of a computer that executes instructions

clustering analysis a process by which a large number of components are grouped into clusters of similar components

Concurrency

A synonym for “parallel processing”, but also applies to single processing environment where multiple programs are interleaved

Confidence level

The probability that a confidence interval actually contains the real mean

Continuous probability distribution

probability distribution associated with a continuous random variable

Continuous random variable

A random variable whose values are uncountable

Data centre infrastructure efficiency (DCIE)

A metric used to determine the energy efficiency of a data centre. The metric, which is expressed as a percentage, is calculated by dividing IT equipment power by total facility power. DCIE was developed by members of the Green Grid, an industry group focused on data centre energy efficiency.

Deadlock

a situation in which two (or more) processes require a resource held by another

Descriptive statistics

Summarize a large amount of data using a small amount of data, often using only one number

E-business

Conducting business over the Internet

E-governement

A generic term that refers to any government functions or processes that are carried out in a digital form over the Internet. Local, state and federal governments essentially set up central websites from which the public (both private citizens and businesses) can find public information, download government forms and contact government representatives.

Efficiency

The speed-up divided by the number of processors

Elapsed time

The total time spent by a job from its submission until its completion

End users

The final or ultimate user of a computer system. The end user is the individual which uses the product after it has been fully developed and marketed. The term is useful because it distinguishes two classes of users, users who require a bug-free and finished product (end users), and users who may use the same product for development purposes.

Extrapolation

A method that infers values from outside the range of values used to build a model

Geometric mean

A measure of location that is applicable to proportions rather than measurements or rates

Harmonic mean

A measure of location that is applicable to rates rather than measurements or proportions

Histogram

A graph which plots the probability distribution

Hit

A resource request for a file from the Web server, as recorded in the server access log.

Hot spot

excessive contention for the same resource (often data in memory)

Independent and Identically Distributed (IID)

A characteristic of a collection of random variables if each random variable has the same distribution and all random variables are mutually independent

Input/Output (I/O)

The process of receiving and transmitting data, as opposed to the actual processing of data

Interactive processing

The processing of tasks with think times in between

Internet

The global set of interconnected networks that use TCP/IP

internet

A collection of packet-switching and broadcast networks that are connected together via routers

Interpolation

A method that infers values from within the range of values used to build a model

intranet

A private internet deployed by an organization for its internal use and necessarily connected to the Internet

Last In, First Out (LIFO)

a scheduling policy that processes tasks in the reverse order that they arrive. Also

called “First In, Last Out”

Latency

The delay imposed by a computing device

Law of large numbers

States that the average of the outcomes of a large number of experiments will approach the expected value

Linear regression

A modelling technique that derives a line equation, relating a dependent data set to an independent data set

Load balancing

The distribution of work to resources so that the loads are relatively equal

Local Area Network (LAN)

a network intended to serve a small area

Loosely-coupled multiprocessor

A multiprocessor where accesses to memory locations can differ, depending on whether the memory is local to the processor or remote mean typically shorthand for “arithmetic mean”

Mean Time Between Failures (MTBF)

The amount of time a component is expected to work without a failure

Mean Time To Repair (MTTR)

the average repair time to fix or replace a failed component and start using the system again

Median

The middle value in an ordered set of values when there is an odd number of values, and the average of the middle two values when there is an even number of values

Memory hierarchy

layers of memory devices, where lower layers have higher capacity but slower access times

Memorylessness

a property of a random distribution such that the probability for an event is not conditional upon the existence of a previous event. A property of a process such that transitions to future states are not dependent upon past states, but only the present state

Mode

the most frequent value in the set of values

Model

An abstraction of a system, often simplifying the details

Multiprocessor

A parallel computer where the processors share resources (usually memory and the network)

Natural (work)load

Characteristics are similar to those of real workload and can be applied repeatedly in a control manner, is developed and used for studies.

Network Attached Storage (NAS)

specialized file servers that serve file system data over a network

Node

An element of a graph or a processor in a multiprocessor or multicomputer

Page depth

The number of unique page views during a visit.

Page impression

The exact number of times a specific website has been accessed or viewed by a user. A page impression acts as a counter for Web pages, informing site owners how many times their sites were visited. Page impressions are also referred to as hits.

Page view

A resource request for a file that is a Web page (e.g., .php or .html files).

Parallel processing

The simultaneous execution of operations

Parallelism

a synonym for “parallel processing”

Percentile

A value in an ordered set of values below which a certain percentage of values fall

Performability

It metrics quantify the system’s ability to perform in the presence of faults. It combines performance and reliability to quantify the operational quality of a service between the occurrence of an error and its full recovery, or over the complete execution. (Eusgeld, Happe, Limbourg, Rohr, & Salfner, 2008)

Performance

It is about the amount of time that an individual transaction or piece of work takes to be completed.

Performance engineering

It: 1) develops practical strategies that help predict the level of performance and 2) provides recommendations to realize the optimal performance level

Performance model

a system’s representation used for predicting the values of performance measures of the system

Pipelining

The simultaneous execution of different stages (or, phases) of an operation

Precision

the amount of scatter in a set of measurements

Predictive statistics

summarize a large amount of data using a small amount of data, but this summary often comes in the form of an equation (also called a model)

Probability

A mathematical expression of the likelihood of an event of an experiment occurring on a scale of 0 to 1

Quality of Experience

Sometimes also known as "Quality of User Experience," is a subjective measure of a customer's experiences with a vendor.

Quality of Service (QoS)

The properties of a network that contribute to the degree of satisfaction that users perceive, relative to the network's performance. Four categories that are considered: (1) capacity or data rate, (2) latency or delay, (3) jitter, and (4) traffic loss

Random

The unpredictability of future behaviour. This is not non-determinism

Random variable

a function that assigns values to the outcomes

Range

the maximum value minus the minimum value in a set of values

Real (work)load

Observed on a system being used for normal operations. It cannot be repeated as such, and therefore, is generally not suitable for use as a test workload.

Regression

A mathematical model derived from measured values

Reliability

Measures of the occurrence of failures during the processing of services

Residence

Time total time spent by a request at a resource

Resolution

The smallest incremental change that can be detected and displayed by a measuring tool

Response time

Time from when a customer arrives to a system until the customer completes service and exits the system

Rule Of Thumb (ROT)

A method of procedure based on experience and common sense

Scalability

System's ability to gracefully increase its capacity in order to accommodate future growth. Scalability can be measured as a ratio of the increase in the system performance relative to the amount of new hardware and/or system software that one have added.

Scheduling policies

Policies responsible for assigning work to be executed over time in order to reach certain objectives, such as minimizing average response time or maximizing throughput

Service Level Agreement (SLA)

A contract between the service provider and the customer. it sets specific goals for response time, throughput, etc.

Service time

Wall clock time between the start of an event and the last byte of output retrieved (typically does not include time to render output on a display device.)

Session

The session of activity that a user with a unique IP address spends on a website during a specified period of time. The number of user sessions on a site is used in measuring the amount of traffic a website gets. The site administrator determines what the time frame of a user session will be (e.g., 30 minutes). If the visitor comes back to the site within that time period, it is still considered one user session because any number of visits within that 30 minutes will only count as one session. If the visitor returns to the site after the allotted time period has expired, say an hour from the initial visit, then it is counted as a separate user session.

Simple linear correlation coefficient

A measure of association which computes the linear relationship between one data set and the other

Simulation

A modelling technique that uses a program to represent relationships between modelled system components. Time is often a parameter

Software Performance Engineering (SPE)

A process of constructing software systems that meet performance objectives

Spatial locality

A workload characteristic of a data stream in which successive references have close addresses. Caching has better performance when spatial locality is present

Static model

A model where time is not a variable in the model

Statistic

A numerical quantity (e.g., mean) calculated from data

Statistical randomness

A property of a sequence of random numbers such that the sequence contains no recognizable pattern or regularities

Statistical regularity

The characteristic of random events to converge to predictable values when an experiment is repeated a large number of times

Statistics

A range of techniques for analysing data, interpreting data, displaying data, and making decisions based on data

stochastic process

A collection of random variables that are parametrized on time

Storage Area Network (SAN)

Specialized networks for storage data that can connect multiple hosts to multiple storage devices

Summary

Statistics group multiple descriptive statistics together to describe data more thoroughly

Synthetic model

A model that is constructed using basic components of the real workload

Synthetic (work)load

Workload generated by synthetic testing tools. Normally used for comparison of hardware.

Systematic error

Errors in measurements that are a result of some experimental “mistake”, such as a change in the experimental environment of an incorrect procedure, that introduces a constant or slowly changing bias into the measurements

Temporal locality

A workload characteristic of a data stream in which the references that occur together in some time period are likely to occur together again in future time periods

Test (work)load

Any workload used in performance studies. A test load can be real or synthetic.

Think time

Elapsed time between the receipt of a reply and the generation of a new request in a closed system

Thrashing

Excessive page file activity due to the lack of adequate memory

Throughput

Rate at which work is executed

Tightly-coupled multiprocessor

A multiprocessor where accesses to all memory locations take the same amount

of time

Time to first byte (TTFB)

The duration from the virtual user making an HTTP request to the first byte of the page being received by the browser. This time is made up of the socket connection time, the time taken to send the HTTP request and the time to take to get the first byte of the page.

Time to last byte

Service time.

Trace-based workload

Workload generation based on the replay of workload parameters that are captured during an earlier execution of a real system

Traffic volume

The number of bytes transferred by the Web server. For example, traffic volume per day is the total bytes transferred to all visitors during a 24-hour period.

Topology

Specifies the connectivity of the processors or computers

Utilization

A number between 0 and 1 showing the average fraction of the total time that a resource is busy

Validation

Determining how close the results of a model are to what would be produced by an actual system

Verification

Determining whether a model is implemented correctly

Virtual Local Area Network (VLAN)

Remotely connected lans that appear to be one

Visit

A series of resource requests from a unique visitor who are temporally clustered. After 30 minutes of inactivity, a page view by the same visitor is counted as a new visit. Visits are sometimes referred to as sessions.

Visit duration

The duration of a visit (i.e., an amount of time a visitor spends browsing through the site during a visit). Visit duration is also known as session duration.

Visitor

A unique client (IP) generating a hit or page view.

Web

Nickname for the world wide web

Weighted statistic

A product of the statistic for a set of values and a weight assigned to it

Wide Area Network (WAN)

A network that covers a large area

Workload characterization

The process of partitioning the workload into smaller sets, where elements of each set are similar in some way

World Wide Web (WWW)

A client/server architecture that integrates various types of information on the Internet

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-2929-0
ISSN 1459-2045