WORKLOAD AND PERFORMANCE AWARE CAPACITY PLANNING

W. A. D. Frank Prabath

(158239B)

Degree of Master Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2017

WORKLOAD AND PERFORMANCE AWARE CAPACITY PLANNING

W. A. D. Frank Prabath

(158239B)

Thesis submitted in partial fulfilment of the requirements for the degree of MSc in Computer Science specializing in Cloud Computing

Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka

June 2017

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic or another medium. I retain the right to use this content in whole or part in future works. (such as articles or books).

Signature: ..... Date: .....

The above candidate has carried out research for the Masters dissertation under my supervision.

Signature of the supervisor: Date:

.....

Name: Dr. H.M.N. Dilum Bandara

ABSTRACT

Given a set of workloads and a performance target, there is a greater difficulty in predicting the exact resource requirement prior to the execution of tasks. While over-provisioning of resources is widely used, it causes a waste due to unused resources. Under provisioning is risky with the possibility of poor performance, user frustration, and SLA violations. While several models are proposed to determine the load, a system could handle for the given set of resources and workloads, determining the required resources to satisfy the expected workload and performance is a long-standing problem. The “Pay as You Go” model in Cloud computing addresses this issue from a different angle. Even then, users do not have a clear idea about the upper bound of their resource requirement, which is essential for financial planning. Moreover, such knowledge is essential in private clouds as the pool of hardware resources needs to be determined a priority based on resource requirement of each application. Such knowledge is also essential in self-hosting.

We propose a model to predict the resource requirement given a workload mix and a performance target. We especially focus on 3-tier web applications hosted on cloud environments. First, a selected 3-tier web application is executed with different mixes of workloads on virtual machines of different capacities. Then the resulting latency is measured. This dataset is then used to build a model using machine learning to predict the resource requirement given a different workload mix and a performance target. The same model is also used to predict resource requirements of other 3-tier web applications. We tested several machine-learning models for their ability to predict the capacity requirement, and random forest gave the highest accuracy. The results showed an accuracy of 97.1% for the same application and 77% for the other application we used as the sample. These results show that the capacity can be reasonably estimated based on the proposed model.

ACKNOWLEDGMENTS

I would like to express profound gratitude to my supervisor, Dr. Dilum Bandara, for his invaluable support, encouragement, supervision and useful suggestions throughout this research work. His continuous guidance enabled me to complete my work successfully.

I am also grateful for the support, motivation, and advice given by Dr. Malaka Walpola and Dr. Indika Perera, by encouraging continuing this research until the end.

I am deeply grateful to my parents for their love and support throughout my life. I also wish to thank my brother, who supported me throughout my work. Finally, I wish to express my gratitude to all my colleagues at 99X Technology, for the support given me to manage my MSc research work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMI	Amazon Machine Image
ANN	Artificial Neural Network
AWS	Amazon Web Services
CIDR	Classless Inter Domain Routing
DB	Database
DOS	Denial of Service
EC2	Elastic Compute Cloud
FLOPS	Floating Point Operations per Second
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
IT	Information Technology
JSON	JavaScript Object Notation
KB	Kilo Bytes
MIPS	Millions of Instructions per Second
ML	Machine Learning
OLTP	On Line Transaction Processing
PaaS	Platform as a Service
PPS	Packets per Second
QoS	Quality of Service
RDMS	Relational Database Management System
RL	Reinforcement Learning
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SQL	Structured Query Language
SVM	Support Vector Machines
TPS	Transactions per Seconds
VM	Virtual Machine
VPC	Virtual Private Cloud
WIPS	Web Interactions per Second

Chapter 1

INTRODUCTION

Every system or an application needs sufficient resources for the proper execution. Most commonly used approach towards the capacity planning was adding required resources as and when there was a need or a demand. Adding an additional processor, memory, or storage, splitting the processing into multiple processing units, or tuning up applications to handle an additional load were the common approaches [1].

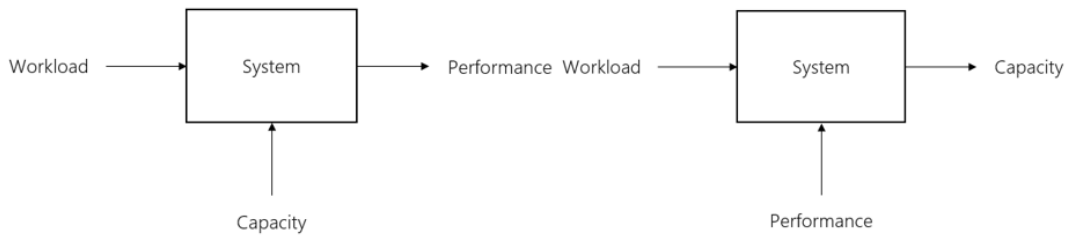
There exist several models and approaches to determine the performance when the workload and capacity are given. Yet, determining the capacity of the system when the performance and workload are given is a long-standing problem. It is extremely difficult to calculate the resource need for a given workload and performance target due to many complexities. Applications are of different nature catering to different needs of the users. The pieces of these applications must be placed in a single container or multiple containers. Applications are required to have the availability need, which adds more pressure to the Dev Ops teams as a demand to the applications will get varied. Applications are required to maintain Service Level Agreements (SLAs) to make sure users will not get dissatisfied while using the applications. Determining the exact need of the resources required to execute applications is the long-standing problem.

Researchers have suggested many alternatives to overcome this issue. The solution provided by Cloud vendors is not an exact solution to the original problem, but a different approach. The “Pay as you go” is a solution to provide resources when required rather than the upfront focusing on the resources required. Most vendors do not have the luxury of keeping resource pools and hence providing the resources whenever requested. Therefore, it is imperative to be able to determine/estimate the required resources. Finding an approach to determine the required capacity in early phases of application development enables better financial planning, reduce project risks, save developer time, and enables developers to focus more on tuning the application to optimize more on the capacity.

1.1 Problem Statement

The proposed research plans to address this problem where required resources/capacity could be planned based on the expected workload and performance needs. Our focus is mainly on 3-tier web applications, as they are very common and would have a wider impact if the problem is addressed. However, making such estimates is nontrivial given that there are vastly different web-applications with different performance targets, workloads, resource requirements (computational, storage, and bandwidth), and built using a variety of platforms, tools, and libraries.

Figure 1.1(a) shows the typical model for performance estimation of a system for known capacity and workload. For example, in [2] researchers came up with a technique to forecast the performance of co-hosted applications via measuring the application's response to different levels of pressure on the shared memory subsystem and measuring the pressure on the memory subsystem [2].



(a) (b)

Figure 1.1 Relationship among the workload, capacity and performance

Figure 1.1(b) shows the estimation of the capacity of a system based on the known performance target and known workload. This is required in several cases. For example, identifying the resources need in Public Cloud is still an important aspect to have an initial idea of the needs of the application and to perform proper budgeting, even though Pay as you go model does not affect the performance. Private Clouds and hosted applications too need to know the capacity needs and the resource requirements for proper planning to make sure required resources will be available when there is a need. Therefore, the problem this research plan to address can be formulated as follows:

How to estimate capacity requirement of a three-tier web application given the workload and performance targets?

1.2 Objectives

This research addresses the above problem based on the following set of objectives:

- To understand the behavioral pattern of 3-tier web applications under different load and resource capacities. Raw data to be classified under varying workloads and performance. The latency will be considered as the performance, while the load to the Application server of the 3-tier web application will be considered as the workload.
- To develop a model to capture the relationship between workload, performance, and capacity, System's behavior is to be modelled using Machine Learning. The initial model needs to be incrementally modified to reach to an accurate model, which will provide results closer to the actual results.
- To validate and demonstrate the utility of the proposed model using appropriate simulation and experimental techniques.

1.3 Outline

The rest of the thesis is structured as follows. Chapter 2 focuses on performance and workload in general, capacity planning and meeting SLAs for performance and then to a detailed discussion on traditional process-based approaches to the model based and rule based models. Chapter 3 presents the research methodology for achieving the objective of capacity planning with the focus on workload and performance. Chapter 4 presents the evaluation of the proposed approach using a load test results of a 3-tier web application. We discuss the process of selecting training and test data to obtain a model and then the verification of the model against a set of test data from a different application. Chapter 5 summarizes the findings, the limitations associated with the proposed technique, and the suggestions for the future work.

Chapter 2

LITERATURE REVIEW

This chapter focuses on the ideas on workload, performance and various parameters to measure workload and performance. Capacity planning, its impact to the business and consequences of not meeting QoS will be discussed later. The traditional process-based approach towards the capacity planning is discussed in Section 2.3.1. Model-based (Section 2.3.3) and rule-based approaches (Section 2.3.4), two major branches of capacity planning are discussed. Latest ideas on focusing the capacity during the application design rather than during the process of deployment is another key area discussed under capacity calculation. In Section 2.4 we discuss the policies associated with allocating VMs. The ways in which capacity is planned against the performance using log-based analysis is covered in Section 2.5.1. How benefits could be obtained via adaptive provisioning is discussed in Section 2.5.2.

2.1 Workload and Workload Measurements

The *workload* is the amount of work an application, host, or a Virtual Machine (VM) has to do. Workload can also be classified as quantitative (the amount of work to be done) or qualitatively (the difficulty of the work). The parameters listed in Table 2.1 should be evaluated to measure the load against the system. When the workload measurements parameters are concerned, processing oriented transaction represents the load given to the processor for processing, Input and output oriented transactions represent the load associated with handling input and output during the processing to capture and send data. The size of the disk and the space available in the disk. The requirement associated with responding back to user, backing up, printing and network connections are needed to be considered. The number of concurrent users for the application and their think time have also played a major role as think time can vary from user to user. Understanding of the time duration in which shows an intensity level for the workload, what makes those intensity levels, monitoring the systems to detect unusual patterns and the judgement of whether system has exposed to security vulnerabilities also known as Denial of Service (DOS) attack should also be considered. 3-tier web application architecture, which consists of presentation,

application, and data tiers have different workloads, such as number of page views per second, HTTP requests made per second, and the number of bytes transferred per second.

Table 2.1 Workload measurements parameters.

Parameter
Processing Oriented Transactions
Input/output Oriented Transactions
Disk space Requirement
Response Time
Backup Requirement
Print Requirement
Network Requirement
Concurrent Users
Think Time

2.2 Performance and Performance Measurements

Computer *performance* is characterized by the amount of useful work accomplished by a computer system or computer network compared to the time and resources used. Performance is measured based on both the internal and external factors [3]. Internal factors that help in diagnosis of performance failures are bottleneck detection (utilization of processors, storage devices, and networks) and number of requests waiting in the various software and hardware queues. Table 2.2 lists a set of factors to identify the performance. Measurement of user-perceived satisfaction and statistics are considered as the external performance metrics. Table 2.3 lists metrics affecting the external performance.

2.3 Capacity Planning

Capacity planning is a process to predict the types, quantities, and timing of critical resource capacities that are needed within an infrastructure to meet accurately forecasted workloads [1]. The main ingredients leading towards the capacity planning are:

1. Type of resources required (processing power, storage, memory, power associated with accessing input and out, etc.),

2. Quantities available of the resources in need,
3. The availability of the resources at the right time and
4. The upfront thinking and the decisions taken on the capacity based on the workload predictions.

The success of the capacity planning is the allocation of the right resources at the right quantities at the right time, avoiding both over allocation and under allocation. Upfront planning and allocation of resources have considered as an art for a long time. The experience and expertise of the experts on the computer systems, extensive knowledge of the domain, empathically understanding of the user personas, and the behavior of the users was considered as the dominant factors of the capacity predictions [4].

Table 2.2 Different performance metrics.

Metric	Description
Throughput	Total number of messages processed in each time interval.
Transactions per second (TPS)	The number of atomic Transactions processed in a unit time. This is an average value and it does not necessarily to be the distributed throughout the processing.
Work done per transaction	Any request to perform a transaction will trigger few more dependent sub tasks. Requesting service calls with external systems, DB accesses, complex processing associated with schema, etc.
Think time	The delays from users when responding to the questions generated by the applications. Throughput does have a direct relation to the think time, based on the transaction whether it is machine to machine or machine to human
Concurrent users	Any system has a user base using the system. A subset of all the users is considered as active users, who is connected to the system. The Concurrent users use the resources of the system at a given time. The number of concurrent users a system supports is the key to any system.
Message size	The larger the message size will lower the performance
Latency	Additional delays associated when a transaction triggers external system calls
Non-functional Requirements	Support towards additional non-functional parameters makes the system complex such as Impact with secure delivery, Impact with availability and up time

Table 2.3 External performance metrics.

Metric
User's Geographical Location
Bandwidth of the Internet connection
Time of the Day
Performance Characteristics of the User's local machine
Probability that requests are rejected

Organizations will lose valuable income, and its public image could be put at a great risk, in situations where users of the systems experience a great difficulty in using those systems. The Quality of Service (QoS) parameters are often set at the deployment stage, rather than at designing phases, which makes System designers and analysts not to worry too much about taking QoS requirements into account when designing and/or analyzing computer systems. Lack of awareness about the issues that affect performance and the lack of a framework to reason about performance are other key reasons for the less focus on QoS parameters, which often neglects sizing a database service, analysis of a datacenter cost and availability, sizing of an e-business based on SOA, and improper allocation of resources to a web service. The QoS attributes of an IT system that are very much important to set the expectations of its users. Not all the systems can be measured against the same set of QoS attributes, even though response time, throughput, availability, reliability, security, scalability, and extensibility are most common.

The discussion starts with a closer look at a process based capacity planning, in which a manual process is used with the delegation of the process to a team in which team will be responsible for monitoring and collecting data and then to make the necessary forecasting. The focus towards the on the fly capacity planning will be the analysis on the relationship between the concurrent demand and the availability of the resources. Model based capacity planning is mostly towards the modelling the given system based on analytic and simulation models, which heavily requires the knowledge about the domain. Rule based capacity-planning focuses on the usage of Machine Language and the Fuzzy logic in determining the capacity.

2.3.1 Process-Based Capacity Planning

Most of the IT infrastructures do not undergo proper capacity planning. Application development does not have a much of a focus on the environment in which the system should be deployed and the load on the system, until users complaining against it. A reactive approach for this is a manual process involves with a set of activities by delegating the process of capacity planning with a dedicated team and formulate a process as shown below, which has multiple steps, focusing more towards the very close monitoring and measuring of the performance of the system [4]:

1. Nominating the process owner for the capacity planning.
2. Identifying the key resources required to be measured.
3. Measuring the utilizations or performance of the resources.
4. Comparing the utilizations to maximum capacities.
5. Collecting the workload forecasts from developers and users.
6. Transforming the workload forecasts into IT resource requirements.
7. Mapping the requirements onto the existing utilizations.
8. Predicting the time when the environment will be out of capacity.
9. Updating the forecasts and utilizations.

Nominating a process owner and empowering that role to coordinate with all the other stakeholders is very critical. End users of the system are unable to predict the future needs and even if the end users are capable of predicting, lack of experience, skills and tool prevent doing so. Reluctance from the capacity planners to use the proper tools, rapid changes in the overall IT direction of the companies, messing up with the planning with managing, and always focusing on important but not urgent activities like budget planning, without the focus on the important-urgent activity of technical planning. Process owner needs to bridge a relation with stakeholders to get everything required to make a better judgment on capacity planning.

Preparing the resource matrix as in Table 2.4 with the combination of the resources categorized into types, configuration details quantity the base requirement, utilization of the resources at the maximum capacity and the availability of the excess capacity to get a good understanding of the resources is required [4]. The forecasted workload need will be taken from the developers and the end users, which needs to be transformed to the IT resource requirements. Resource requirement is mapped to

utilizations, which helps to predict out of capacity and to forecast the utilization need for the future [1].

Table 2.4 A sample resources matrix.

Resource Type	Configurations	Quantity	Base Requirement	Utilization to Maximum Capacity	Excess Capacity
Processor					
Memory					
Disk space					
Network bandwidth					

Performance and throughput expectations from the different systems are calculated based on different measurements and metrics. As per the Table 2.5, which shows different measurements considered in different systems and components, it is obvious that determining performance involve careful consideration of multiple metrics.

Table 2.5 QoS attributes of different systems.

System	Throughput Metric
OLTP System	Transactions per Second (tps)
Web Site	HTTP requests/sec
	Page Views per Second
	Bytes/sec
E-commerce Site	Web Interactions Per Second (WIPS)
	Sessions per Second
	Searches per Second
Router	Packets per Second (PPS)
	MB transferred per Second
CPU	Millions of Instructions per Second (MIPS)
	Floating Point Operations per Second (FLOPS)
Disk	I/Os per Second
	KB transferred per Second
E-mail Server	Messages Sent Per Second

Performance is the major consideration in capacity planning, which spans across many concerns, factors, and methodologies. Determining the performance involves with careful consideration of many concerns, related to latency, throughput, capacity and modes having a very high dependability with the expectations from the system in the given scenario. The demand for the application and the system in which the application has been deployed are two major factors affecting the performance. Even though, the resources allocated are lower, the application would meet the expected performance when the demand for the application is also lower. The analysis methods have been very widely used to find the performance of the systems. Figure 2.1 shows these concerns, factors, and methods associated with determining the performance [5].

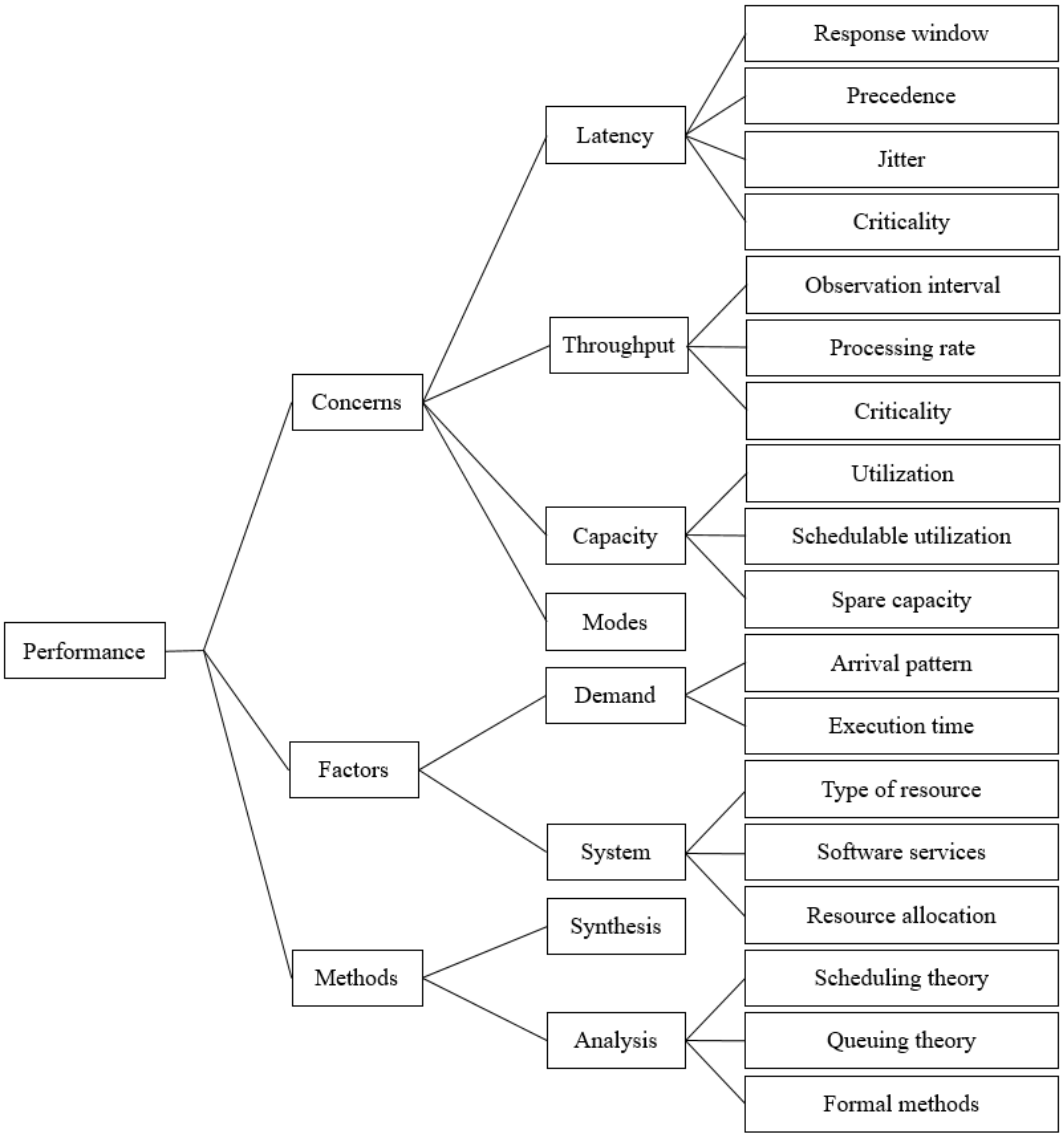


Figure 2.1 Concerns, factors, and methods of performance

2.3.2 On the Fly Capacity Planning

Capacity planning is the art in which based on the relationship between concurrent demand and resource availability. Performance issues cannot be easily solved in an environment when deployment topology of applications is complex. When the code base is considered as a black box and performance is getting narrow down to its simplest form, adding additional resources may solve performance issues, yet it is not the best approach. An approach that closely monitors the code execution to take decisions on the call stack sampling (contains execution state, information about monitors in running applications) and the resource utilization feeds from the operating system of an already running system gives a better understanding about the system. A simulation of tuning actions (change to the code or the environment in which the application runs) based on the changes in the resource demand spanning over resource and code changes gives a reasonable figure for the on-the-fly capacity planning.

Latent bottleneck causing from shifting the resource demand, zero-sum games where additional demand, causing the existing processes to run slowly and head fakes in which the inability to determine which process causing the performance issues are few daemons associated with performance tuning.

Analysis of the stats collected from different resources and the way systems has performed when the resource demand is getting shifted is the core process associated with on-the-fly capacity planning, with the differentiation of focusing on tuning actions rather than the arrival of work [1].

The effectiveness of modelling tuning plans (an unsorted set of tuning actions with the implication that their impact on performance be considered as a unit) is higher when the concurrent demand and concurrent capacity is observed over the arrival rate and the service time of the resources. There is a continuous set of work requests to the server, which makes the demand for the resources required as a function of the rate at which requests are made and the time the request happened. As per the Little's Law, the demand is a product of arrival rate and the service time, which is a very important point in determining the demand [1].

There is a possibility that concurrent demand to exceed the concurrent capacity of a resource. This will create a backlog to be served which multiplexed resources or gating (e.g., locks) in such a way to tolerate concurrent demands.

2.3.3 Model-Based Capacity Planning

System modelling is carried out based on measured control inputs and control outputs. Model-based approaches extensively require the domain knowledge [4]. Even though, it is difficult to model a complex system. Model-based approaches provide more internal details, which makes it easier to understand the functioning of systems. Model-based approaches have following advantages:

- Analytic models are less expensive to construct and tend to be computationally more efficient to run than simulation models.
- Because of their higher level of abstraction, obtaining the values of the input parameters in analytic models is simpler than in simulation models.
- Simulation models can be made as detailed as needed and can be more accurate than analytic models.
- There are some system behaviors that analytic models cannot (or very poorly) capture, thus necessitating the need for simulation.

Linear regression is also used to model the system as a black box under the control theory based approaches. Control theory and queueing model are the major subclasses of this family. Careful monitoring on demand and the responses from the application to the demand variations plays a key role in finding the requirement for the resources. Controlling the admission involves careful study of queueing theory and the domain in which the applications are running. The component VM scaling has the responsibility of dynamically allocating the resources. Table 2.6 discusses components of the model and the responsibilities associated with the major components [3]. This is a good measure to show the major steps to come up with capacity planning based on a model, which starts from the identification of the relation between the control input and output to the controlling and scaling based on the computing resources.

Model-based capacity planning has several more issues, which makes them a solution in an academic world rather than providing benefits to an actual Business scenario.

The open issues from model-based approached on capability planning could be categorized as follows in Table 2.7.

Table 2.6 Components used for resource allocation.

Monitor	Admission Control	VM Scaling
Identifies the relation between control input and control output	Applies both queueing theory and linear regression. Queueing theory requires more of a domain knowledge. Linear regression is having the challenge of defining the relationship among control variables.	Allocates resources dynamically for computing resources. Vertical scaling involves changing the size of a VM instance Horizontal scaling involves adding or removing VM instances

Table 2.7 Open issues in model-based planning.

Open Issues in Current Model Based Approaches	Open Issues in New Trends for Model Based Approaches
There exists a different type of models and it is not very much clear on which model type to be used as there does not exist much of details on the advantages and limitations on each model.	Even though most of the new studies have been carried out using linear methods, settings of QoS targets heavily depends upon the non-linear.
It is difficult to set an appropriate controls inputs to determine the out controls as the relationships between controls for input and output varies significantly when workload regions get changed.	Even though nonlinear controls need simpler implementations, quick response and less cost involves, it requires a rigorous mathematical analysis, which is a as a big disadvantage.
Inability to determine the feasibility of control inputs since for certain output controls, there exists negative values for input controls.	Resource allocation for distributed web applications is not simply due to the complexity of systems and the nature of resource sharing in a cloud environment.
It is difficult to ensure the robustness to the changing of the workload due to the contribution of errors. When the error contribution is getting larger, it has a bigger impact to the performance.	The dawn on newer web applications and introductions of powerful client side technologies has shifted the load in to client browsers from the servers.

2.3.4 Rule-Based Capacity Planning

Rule-based approaches help to take decisions where uncertainty plays a major role. Machine Learning (ML) and Fuzzy control are the two major categories in rule-based capacity allocation [4]. ML needs to go through a vast array of historical data related to resource utilization and performance metrics for an efficient resource management. The RL offers the ability to enhance the allocation policies without the model knowledge by learning. The learning process of RL is such that the learning agent

learns to make accurate decisions with the interactions of the external systems, as RL learns policies in dynamic environments based on finite MDP. SVM is widely used in areas such as pattern recognition, classification, and data mining, yet are not suitable for resource management of on-line applications due to the time complexity. ANN is used to predict the resource demand for VM scaling. ML algorithms are not a good candidate for resource management due to the performance issues, even though they do not need much of a domain knowledge [4].

Fuzzy control approaches, on the other hand, are easy to implement and manage, as those are governed by a set of predefined rules, which makes themselves a good candidate for resource management in multi-tier systems. Falsification, inference engine, and defuzzification are the main stages of its decision-making process. Unlike ML algorithms, Fuzzy control approaches ensure performance guarantees as well. However, it is required to conduct sufficient simulations to design a fuzzy control system to achieve optimum performance [4].

The neediest of less domain knowledge and ability to learn from historical data are the major advantages of rule-based approaches, which further requires a many number of configuration parameters after running many simulations on different scenarios. Lack of governing QoS parameters causes another challenge when using the rule-based approaches to solve issues in resource allocations. Rule-based approach is the appropriate and most suitable way to move forward as it has a more practical valid reason. Table 2.8 summarizes the advantages and disadvantages of the rule and model based planning.

2.3.5 Capacity Calculation

Another approach towards the capacity planning focuses on measuring the performance values for a set of defined parameters, which are not perceptually based and focuses on designating applications to perform better in the given environment. Rather than focusing on tuning the application at the deployment stage, a much more focus is given during the application design phase. All the possible optimization steps are taken into the consideration. Usage of a caching strategy, usage of lower memory and processor consumption, effective use of shared resources, proper usage of

configuration values, thorough testing and profiling are major aspects to focus before the application goes to the production [6].

Table 2.8 Advantages/disadvantages of rule and model based planning.

	Rule Based	**Model Based**
Advantages	Less Domain knowledge	Depends heavily on Domain knowledge
	Based on Mathematical	Based on Mathematical
	Rules are derived from historical data	Provides more insights
	Keep QoS guarantees	Issues with resource allocations could be overcome by using control theory
Disadvantages	Requires lots of simulations and training	Difficult to model a complex system using a mathematical model.

Load balancing and routing, clustering, state replication, auto-scaling systems, and auto healing systems are some architectural concerns considered which will can scale (both scale up and scale out). Disaster recovery, backup and recovery are other aspects, which draws the attention to make the application available in any mode such as cold standby, warm standby, hot-standby, and active-active.

Usage of monitoring tools and the type of hardware (physical, VM, and Cloud based) to be used in the production environment are the other considerations. Additional allocation of resources required to support the peak demand has also to be forecasted for a period, while downtime is to be agreed.

Once the high-level solution architecture is defined with performance in mind, collecting capacity planning data, finding the requirement of the capability planning, identifying non-functional requirements, and setting the benchmark for the performance makes demands to identify the instance counts are required. These require long-running performance test to conclude the architecture for the deployment, so that capacity data could be gathered for the capacity calculation.

2.4 Policy-Based VM Allocation in IaaS

A special attention must be taken when the resources are allocated in the Cloud, where resources in a pool must be managed. The need to have proper policies in place is very vital for allocation of resources and during the process of placing the VMs, once the computing need has been identified. Immediate, best effort, advanced reservation and deadline sensitive are the most commonly used policies among the Cloud Providers [7].

Cloud providers maintain different types of VMs, providing different QoS and charging schemes to support the different needs of the customers. This has given Cloud providers the flexibility in managing resources and utilize their resources at an optimum level [8]. Table 2.10 lists most commonly used pricing models in Cloud Computing. For example, AWS Elastic Compute Cloud (EC2 [9]) provides spot, reserved and on demand pricing models for their customers.

Cloud Federation [10] has been yet another booming idea on allocating resources in a Cloud environment, which helps to overcome during the high demand for the VMs, by outsourcing the requests to other members of the same federation. Usage pattern, types of requests and infrastructure costs are considered as the three major factors affecting the resource allocation, which helps to focus more on pricing, profit utilization and QoS. Policies as shown in the Table 2.10 are required not only to focus on the profit aspects, but also to make sure service is based on the agreed QoS and SLA so that user satisfaction has also been met [11].

Table 2.9 Pricing models in cloud offerings.

Pricing Model	Characteristics
Reserved	There exists a long-term commitment
On-demand	Long-term commitment is not required Hourly payment
Spot	Lower charges compared to reserved and on-demand Cloud provider can terminate this type and allocate to a higher bid

Table 2.10 Policies for VM allocation.

Policy	Policy Description
Non-Federated Totally In-House	Termination of Spot VMS with lower bids are considered
Federation-Aware Outsourcing Oriented	Request a VM from a member in the Federated group and if not available, Spot VM will be terminated
Federation Aware Profit Oriented	Profit is compared against terminating a VM and outsourcing a VM

The number of requests for the Spot VMs, the impact on the load and the number of members in the federation group are some other facts affecting the results when a policy has been applied. It is also important to have Business focused policies to prevent over-provisioning of IT resources, which optimizes the usage of IT resources. So, that, consumption of physical, energy, and human resources will be very low while increasing efficiency and minimizing IT budgets.

2.5 Performance and Capacity Planning

2.5.1 Log-Based Performance Analysis

Logging is a universal common approach used by the application developers, as an easier way to track the execution flow of functionality in computer systems. The data that get logged varies from system to system and can externally configure on what is being registered. The most common types of log data are the details about the process or the routine, timing details, the people or other sub-systems performed those routines and much more, including the history of routines. An approach called “Process Mining” techniques were used to explore, track and enhance the actual processes using the knowledge from the logs [12].

An approach towards the performance measurement of SOA through log-based analysis requires a wide variety of metrics, including the areas of business, IT pervasiveness, financial, quality management, management, project, corporate, quality and timelines together with technical aspects. When the metrics are accurate and realistic, this will provide the facility to effectively diagnose the root causes associated with the performance problems.

Once the metrics are defined, results should be mapped with the identified performance metrics from the measurement tools. For further analysis, the adaptability of SOA and enhance the performance, performance metrics must be combined with advanced evaluation tools based on the logging capability [12]. Table 2.11 shows the metrics categorized into 3 major areas.

Table 2.11 Different modes of metrics.

Process Metrics	Place Metrics	Activity Metrics
Average Throughput time	Waiting time	Waiting time
Minimum Throughput time	Synchronization time	Execution time
Maximum Throughput time	Sojourn time	Sojourn time

Log files were given to the system for the processing and mining together with the mined-model. The plug-ins have the capability to provide the details to the supporting key decision areas as per the following:

- Routing possibilities
- Average service time for a task
- Average throughput
- Minimum throughput
- Maximum throughput
- The bottleneck in the model
- Time spent in between two processes

2.5.2 Adaptive Provisioning

Cloud Provisioning is the process of deployment and management of applications on Cloud infrastructures, such as VM, resource, and application provisioning (ensuring an efficient utilization of virtualized IT resources). Creating and allocating VMs as resources in Cloud environments for the applications has to overcome the issues related with modeling workload and performance, deployment, monitoring and virtualization techniques as it has to deal with uncertain behaviors and lack of understanding of IT resources and network elements, errors in estimations (arrival pattern, I/O behavior, service time distribution, and network usage) and highly dynamic nature due to the usage by a larger number of users, as well as due to the

variations among different application types (high performance, web hosting, and social networking) [13].

Provisioning techniques having the capability to automatically adopt to the demand by facilitating dynamic resource allocation and satisfying SLAs requirement must be the bare minimum need. Because Cloud vendors do not expose the details on underline hardware, an approach taken from analytical performance and workload (arrival pattern, resource demands) information must provide the necessary information about the requirement to the provisioned, which is a very complex activity as provisioned needs to make sure it calculates the best, optimum and efficient resources which ensure reaching the QoS targets. Adaptive provisioning should also focus on automating routine management tasks, flexibility of assigning virtualized IT resources when it is needed while not over-provisioning and not impacting QoS limits. Several QoS metrics to be considered are as follows:

- Monitored average request execution time
- Application instance queue size
- Expected arrival rate of requests
- Maximum number of VMs allowed

Adaptive provisioning needs collaboration of 3 major components. Performance should be modelled to predict the load, which will be handled by the load predictor, work load should be analyzed for a period by the workload Analyzer and the understanding about the applications by the application provisioner. Table 2.12 shows the components and a detailed description of their main functionality, which performs the adaptive provisioning.

2.6 Supervised Machine Learning

Supervised Machine Learning is achieved via a set of data and making predictions based on that set of data. Features (also called as the observations) are things that might have an impact towards the target we want to predict. What we need to predict is called as the Label, which the resource needs in our research. Supervised machine learning is used to find the correlation between features and the label, based on algorithms. Regression algorithms used to predict the value on the continuum over the series of

values. The model will be created so that the predictions will be able to make on future features.

Table 2.12 Components of adaptive provisioning.

Component	Functionality
Application provisioner	Main points of contact in the system that receives accepted requests and provisions virtual machines and application instances based on the input from workload Analyzer and from load predictor and performance modeller. VM and application provisioning is performed by the application Provisioner component based on the estimated number of application instances calculated by the load predictor and performance modeller: if utilization of data center resources is low, application Provisioner is directed to destroy some application instances.
Workload analyzer	Generates estimation of future demands for the application. This information is passed to the load predictor and performance modeller component.
Load predictor and performance modeler	Solves an analytical model based on the observed system performance and predicted load to decide the number of VM instances that should be allocated to an application Responsible for deciding the number of virtualized application instances required to meet the QoS targets Model parameters are obtained via system monitoring and load prediction models. Efficient mapping of requests, while the goal of VM Provisioning is to provide applications with sufficient computational power, memory, storage, and I/O performance to meet the level of QoS expected by end-users. Responsible for generating estimation (prediction)

As shown in Figure 2.2, supervised learning, it starts from the Dataset in which we need to give it to the Machine Learning. A cleaning of data is the next step in which incomplete data will be removed, missing data will be replaced with default values and formatting the data to suit the need so that the ML algorithm will be able to process the data. A portion of the data (the majority of data) will be selected as the training data and used to train the model. The rest of the data could be used to test the accuracy of the model as we have the data from the system itself to make sure our predictions are accurate by comparing the output from the model with the known dataset we have reserved with the labelled dataset. An algorithm is used to train the model, mainly the correlation between features and label. Linear regression, Neural network regression, boosted decision tree regression are possible algorithms to be considered in regression mode [14]. The score model takes the response came from train model into the

consideration and label dataset is used to determine the values of the train model to compare the values each other.

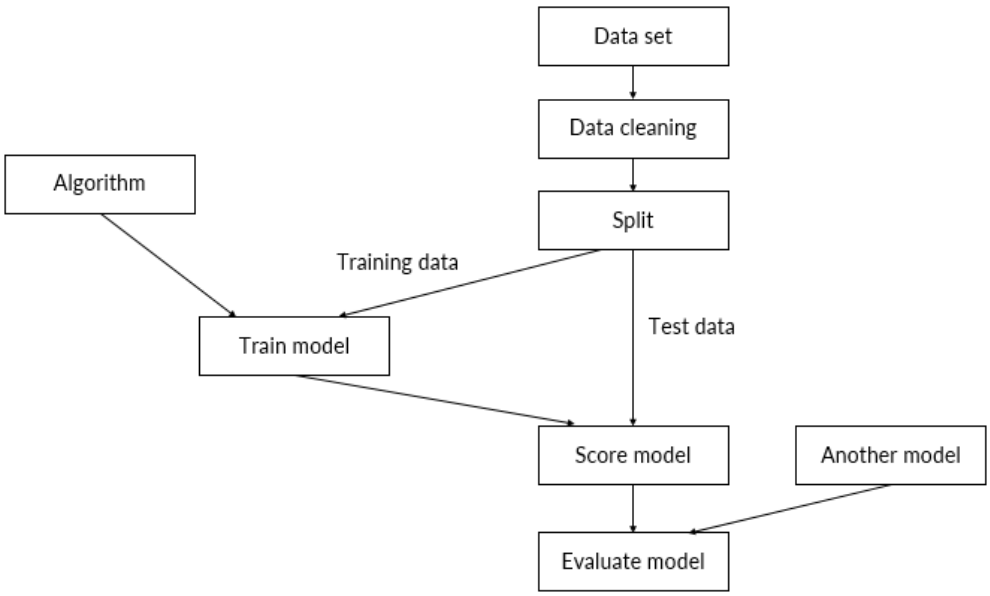


Figure 2.2 Supervised machine learning.

2.7 Summary

In this chapter, we discussed the related work on capacity planning. The process-based approach has little to offer as far as the accuracy of the capacity planning is considered, since it only focuses on a set of steps to allocate resources without any measurements. On-the-fly capacity planning focuses on dynamically allocating resources based on the demand, in which this is not a good approach. Model-based capacity planning needs extensive knowledge about the domain and it is mainly based on simulation models. The capacity calculation is the calculation of the capacity for a set of defined parameters and emphasis more about the capacity needs during the application design phase. Policy-based VM allocation is the most preferred approach in Amazon EC2 to provision VMs based on the availability and the need, in which the exact logic and the allocation on VMs is only known to the Cloud vendors. Log-based performance analysis helps to figure identify areas in which application showed the bottlenecks as far as the performance is considered. Adaptive provision is another approach used by cloud vendors, yet again fully visible only to the cloud vendors. An approach based on the supervised Machine Learning will be chosen as the preferred approach, as there exists a correlation between workload and the capacity towards the system performance.

Chapter 3

METHODOLOGY

This chapter presents the approach and the methodology for building a model to determine the capacity requirement when the workload and the required performance SLAs are given. Section 3.1 discusses how we come up with a model that determines the capacity for the given workload and performance need. We discuss the main steps in the process including the data collection from sample application in Section 3.2, preprocessing data to build the model under Section 3.3, and details of generating the model under Section 3.4. Algorithms we used for the machine-learning approach, selecting the score model, and evaluating the model against another application under Section 3.5.

3.1 Methodology

Machine learning lets us find meaningful, predictive patterns in existing data, then create and use a model that recognizes those patterns in new data [14]. A proper model must be obtained as per Figure 3.1, which shows the steps in reaching a suitable model to capture the relationship between workload, performance, and capacity. Machine learning suits better, as it is based on creating experiments hoping to improve a predictor, whereas traditional process focuses on incrementally building solutions by completing discrete features. Either supervised learning, in which the value to predict will be in the training data itself or unsupervised learning, in which the value to predict will not be in the training data may be used depending on the availability of data and accuracy of the model. Multiple iterations may be required to reach towards an accurate model.

Let w will be the workload mix, p will be performance target, and c be the capacity needed to handle workload w with performance p. w, p, and c are vectors as each capture multiple attributes. Given a set of observations of (w_i, p_i, c_i) our objective is to build a model $\forall i$ f (w_i, p_i, c_i) that captures the relationship between workload, performance target, and capacity requirement. Deriving the function f, such that it will fit the observations is the process of learning. Once the model is derived, it is to be

used to predict the required capacity c_j to handle a given certain workload w_j with performance target p_j.

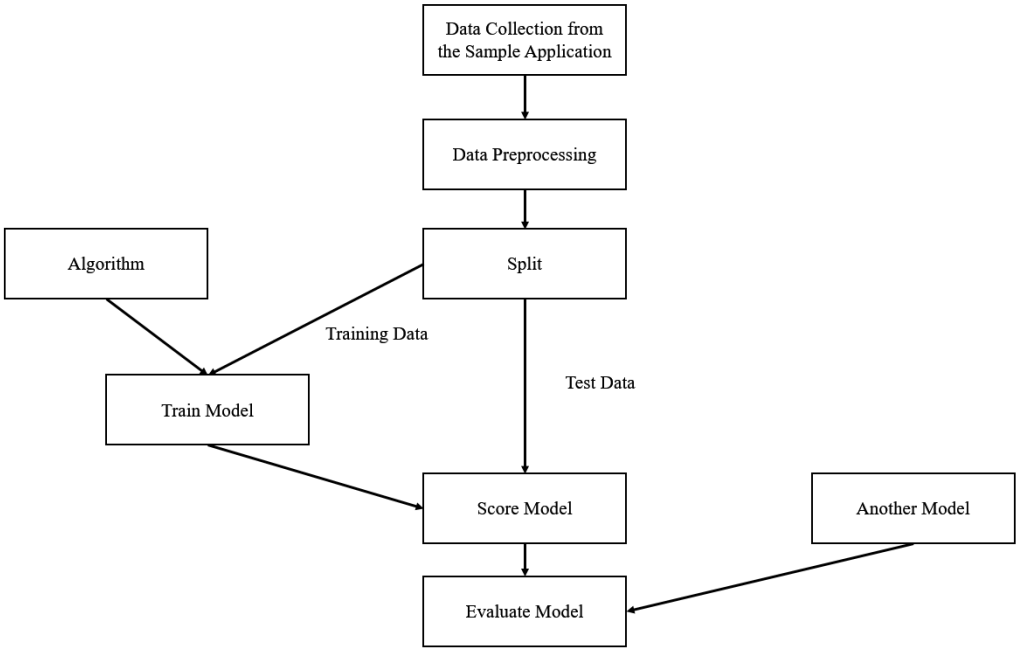


Figure 3.1 Research methodology on finding a suitable model.

Identifying a suitable raw dataset is the starting point, which can be achieved through data collection from a set of sample applications. Then the data need to be pre-processed to make it suitable as an input to the machine learning algorithms to be considered. Parameters of the selected algorithms need to be tuned to get a better model with higher accuracy. Then the selected and optimized models need to be tested for other applications to identify the most suitable model. Once we can predict the capacity for the performance and workload at a higher accuracy, we select it as the evaluated model. Then the finally chosen model should be able to predict the capacity for a given workload and performance requirement for other applications with reasonable accuracy.

3.2 Data Collection from Sample Application

The test data need to be collected from a load test against a 3-tier web application that is representative of typical web applications. The database of the 3-tier web application, application server, and the web server were deployed as cloud instances. Application server processes a complex logic to generate the entire web page based on

the configuration settings persisted in the Database. It processes the form controls, handles workflows, and use user security permissions to hide or show user controls on the web page in the form of a JSON file. The Web server sends the generated JSON file to the clients. Application server handles workflows within the same web page and/or with other web pages based on the rules as the core of this selected application. Therefore, as per our observations, the typical resource requirement of the Application server is higher compared to both the Web and Database servers.

Another cloud instance was used to generate the load against the 3-tier application. A dataset that covers a broad spectrum of user scenarios have to be selected to make sure all the components of the application will be covered and a load will be applied not only to the one component but to all the components. This step consumes a lot of time, as we have to execute the same user scenarios (i.e., workloads) against the 3-tier web application deployed on servers/instances of varying capacities. It was easier to simulate varying capacities in a cloud computing environment, as the CPU and memory can be fined tuned while keeping the same application deployment. For example, Amazon EC2 allows keeping the same deployment in a VM and configuring it to be a different type of VM with a different number of virtual CPU cores and memory. Therefore, the application does not need to be re-deployed when the VM instance type got changed.

The collected data is usually in a format, which is easier to run the load test. The easiest and quicker way to run the load test was to have the environment ready for a hardware setup and apply the load for the number of users we wanted to run against. We repeated the same load for the next hardware setup, until we complete the load test across the planned array of hardware configurations and users. For example, we can have the application deployed in one VM and then apply workload while varying the number of concurrent users.

A typical workload of a user consists of different *scenarios*, where scenario consists of different actions/steps such as login to the web application, searching for an entry, modifying and saving that entry, and logout. The latency will be obtained for each user step to make the analysis easier. Hence, it is important to understanding different user steps of the chosen web application using code walk-throughs. Then to have a good

coverage of the application's functionality steps related to database reads, updates, insertion, and joins need to be selected. For example, Table 3.1 shows the relationship between scenarios and individual steps, where the same granular steps will be touched by multiple scenarios. These granular steps give a better indication on the number of database tables and database operations each step is associated with, so that we will have a better gauge on the complexity of each scenarios of our experiment. Then scenarios need to be picked to have a good mix of application steps.

Latency and resource utilization were selected as metrics to measure performance, and each term is defined in Table 3.2.

Once we finalized the scenarios, we collected the data in the format similar to that of Table 3.3. We measured the load by the number of users and the scenarios, capacity by the hardware, and performance as the latency. We collected the data for different capacities and different number of concurrent users. We got a diverse dataset as we used different combination of scenarios, across different capacities against a range of concurrent users.

Table 3.1 Scenarios and steps.

	Scenario 1	**Scenario 2**	**Scenario 3**	**Scenario 4**	**Scenario 5**
Step A	X	X	X	X	X
Step B	X	X	X	X	X
Step C	X				X
Step D	X				X
Step E		X	X		X
Step F	X	X	X	X	X
Step G	X	X	X	X	X
Step H				X	
Step I				X	
Step J	X				
Step K		X			
Step L			X		
Step M				X	
Step N				X	
Step O					X
Step P					X
Step Q	X	X	X	X	X

Table 3.2 Selected performance metrics.

Metric	Description
Latency	Difference in the time between request and response
Resource Utilization	Amount of CPU and Memory required to process

Table 3.3 Data recorded from load testing for 3-tier application.

Scenario	Hardware	Average response time in seconds						
		N1 Users	N2 Users	N3 Users	N4 Users	N5 Users	N6 Users	N7 Users
Scenario_1	Hardware_1	x	x	x	X	X	x	x
Scenario_2	Hardware_2	x	x	x	X	X	x	x
	…	…	…	…	…	…	…	…
Scenario_n	Hardware_n	x	x	x	x	X	x	x

3.3 Data Preprocessing

Our test data was in a form similar to Table 3.1. However, the dataset needs to be transformed to a table format to determine the relationship between the performance, workload, and capacity. Therefore, first, we arrange the data to a suitable format by assigning numerical weights for the steps.

First, we need to map the type of operation, number of users, and latency it takes to perform the operation against the capacity as seen in Table 3.4. We will read the values in this table as the U_1 number of users will take T_1 latency to perform the $Scenario_1$ against the $Capacity_1$, U_2 number of users will take T_2 latency to perform the $Scenario_2$ against the $Capacity_2$, and so on.

Table 3.4 Data arranged for user scenarios for users

Type of Operation	Users	Latency	Hardware
$Scenario_1$	U_1	T_1	$Capacity_1$
$Scenario_2$	U_2	T_2	$Capacity_2$
…	…	…	…
$Scenario_n$	U_n	T_n	$Capacity_n$

We need to find the actual impact to the system from each step associated with a scenario, as these operations represent the workload of our model. We need to convert the complexity of each step/action to a set of numerical values as shown in Table 3.5. As the focus is on 3-tier web applications, we consider the number of database tables touched by a particular step to be representative of the workload, as selected step

retrieve, insert, or update records stored in multiple database tables. We know the number of database tables touched by each step, as we walked through the code of the application. We can also identify the type of database operations for each step.

Once this is known we have two options. The first option is to assign equal values to query, create, and update database operations. We assigned a weight of one for database query, database create, and database update. A weight of two was assigned to varying values through query, create, and update database operations based on their complexity. We assigned the weight of one for the database query, weight higher than one for database create and weight much higher than database create to database update in the second option. Database query operation is the lowest in terms of complexity and performs faster than database create and database update. Database update operation is with the highest complexity in RDMS compared to the database create, as database updates will have to query the records and then to change the values [15]. Table 3.5 shows numerical values for each scenario, which is the sum of complexity of all the steps associated with the scenario. Numerical values are the ideal representation for the scenarios to build the model rather than having text labels. We build Table 3.5 for both the options described above to cover equal weight and varying weight.

Table 3.5 Weight for sub scenarios.

Sub Scenario	Database Tables	Query	Create	Update	Weight for User Scenario
Sub Scenario 1	X	x	X	x	X
Sub Scenario 2	X	x	X	x	X
…	…	…	…	…	…
Sub Scenario n	X	x	X	x	X

Table 3.6 shows the prepared data having all numerical values for the user scenario, latency and users, while hardware remains as the instance type. Such a dataset is useful while applying a machine-learning model.

Table 3.6 Data prepared for the machine-learning model.

Sub Scenario	Hardware	Latency	Users
x	x	x	x
x	x	x	x
…	…	…	…
x	x	x	x

3.4 Model Generation

We extracted 80% of the preprocessed dataset to train the machine learning models while remaining 20% of the dataset was used as the test data. We model the dataset such that capacity as the dependent variable while performance and workload as independent variables. Once the model is created, we used the test data to check the prediction accuracy of the chosen model. We randomly selected the training dataset from the preprocessed dataset to make sure that the training dataset is well distributed.

To build a model we choose machine-learning models due to the following advantages [16]:

- Comparatively more accurate than rules generated by humans
- No need to have the human involvement
- Flexibility in certain models support feature generation for any features
- Ability to explain the data from the automatically generated hypothesis

As our main goal is to automatically produce near accurate predictions based on the data we collected, we choose following machine-learning models:

1. K-Nearest Neighbor (KNN) [17] – Given a workload and a performance target, we can get a reasonable estimate of capacity requirement by analyzing the capacity requirements of similar workloads and performance targets. For this KNN is a suitable model where capacity recommended by k (≥ 1) nearest neighbors based on their workloads and performance can give a reasonable indication of capacity required to handle the given workload and achieve given performance target. In this case, we model our problem as a classification problem where data points are classified as based on their workload and performance, and labeled based on the capacity.
2. Random forest [18] – In random forests, we can model the problem again as a classification problem based on a set of decision trees. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.

   This makes random forests a good option for this experiment, as we want to have a good classification model.

3. Support Vector Machine (SVM) [19] – SVM is also a supervised classification algorithm that identifies the right hyperactive plane by careful classification of data considering the distances to the planes. SVM easily handles feature interactions and non-parametric; hence we do not have to worry about outliers or whether the data is linearly separable. SVM has strong theoretical foundations and excellent empirical successes on classification problems [19].

Each of the above algorithms has a different set of parameters to tune. For example, in KNN value *k* needs to be set. In random forests, we need to set the number of trees. Similarly, in SVM cost value needs to be set.

The key here is to have a model that could accurately determine the capacity for the given performance and workload. We use the test dataset collected from the application to verify the accuracy of the proposed model. As seen in Table 3.7 we used it to cross check the predicted capacity for the given performance and workload values. When the predicted value is same as the actual value, we considered the prediction to be successful, which means that the capacity predicted from the model is same as the value used in real data. We define *accuracy* as the number of capacity predictions that are same as in original data. High accuracy indicates the chosen model is suitable in predicting the capacity requirements, as well as scenario to number mapping is effective in capturing the workload complexity.

Table 3.7 Actual and model predicted values for capacity.

Performance	**Workload**	**Actual Capacity Value**	**Model Predicted Capacity Value**
P1	W1	AV1	MPV1
P2	W2	AV2	MPV2
…	…	…	…
Pn	Wn	AVn	MPVn

A model with the highest accuracy is then picked as the score model as shown in Figure 3.1, which is also considered as the candidate model. In addition, this model will be used to predict the capacity for the given workload and performance requirement for the applications of the same nature.

3.5 Evaluating Prediction Ability of Model for Different Applications

To be generally useful, the most accurate model for the chosen application should also be able to predict the capacity requirements of similar 3-tier web applications. As the model captures the relationship among workload, latency, and resources, we use the model to predict the resource requirement given the workload and performance of chosen application(s). Then the prediction accuracy is checked for the predicted resource requirement against the actual capacity. The model that performs well on the original application and new application is then chosen as the most preferred model for capacity planning.

3.6 Summary

We proposed a methodology to predict the capacity requirement of a 3-tire web application given a workload and a performance target. We set up a modular environment using virtual machines. We then ran a set of load tests and collected the performance of a given combination of workload and resources. Then the dataset is preprocessed to map the workload complexity based on the number of database tables involved and the type of database operations to be performed. Three machine-learning models are then chosen to model and predict the capacity requirement given a workload and a performance target. We then evaluate the prediction accuracy of a given test dataset. The same models are also evaluated based on their ability to predict the resource requirement for another 3-tier web application. Finally, we chose the model that could predict the resource requirement of the other application with the highest accuracy as the chosen model.

Chapter 4

PERFORMANCE EVALUATION

This chapter focuses on performance evaluation of the proposed technique. Section 4.1 focuses on setting up the test environment in the Cloud, while Section 4.2 focuses on evaluating the performance of different machine learning models. Section 4.3 focuses on the accuracy of each algorithm and shows the reason behind selecting the model to find the capacity need for the given performance and workload.

4.1 Experimental Setup

The collect the data and evaluate the model we setup two 3-tier web applications, in which one was from the Health Care domain and other was from E-Commerce where we used the popular Rubis [20] website. Next, we describe the high-level architecture of the deployment topology. Setting up both the web application environments in AWS was the key, as we could emulate a production environment in AWS VMs with appropriate configurations.

4.1.1 Computing Infrastructure Setup

It is critical to have a near-production environment setup to perform the performance/load testing as we are planning to achieve high accuracy (e.g., 80% of the capacity needed to have a performance benchmark against an expected user load). Ideally, performance testing must be performed in the same setup that will be used by the end users when they have the production environment to be used. However, having the same setup in a cloud is not an easy step as performance testing cannot be isolated from the security concerns associated with this deployment topology.

We setup a Virtual Private Cloud (VPC) [21] with four subnets within the VPC as shown in Figure 4.1. We deployed the application using an application server, web server and a database server in their respective subnets to perform the load testing. We also setup a workload generator based on JMeter [22] version 3.2 on a separate VM. Depending on the application used for testing, we changed the workload generated by JMeter.

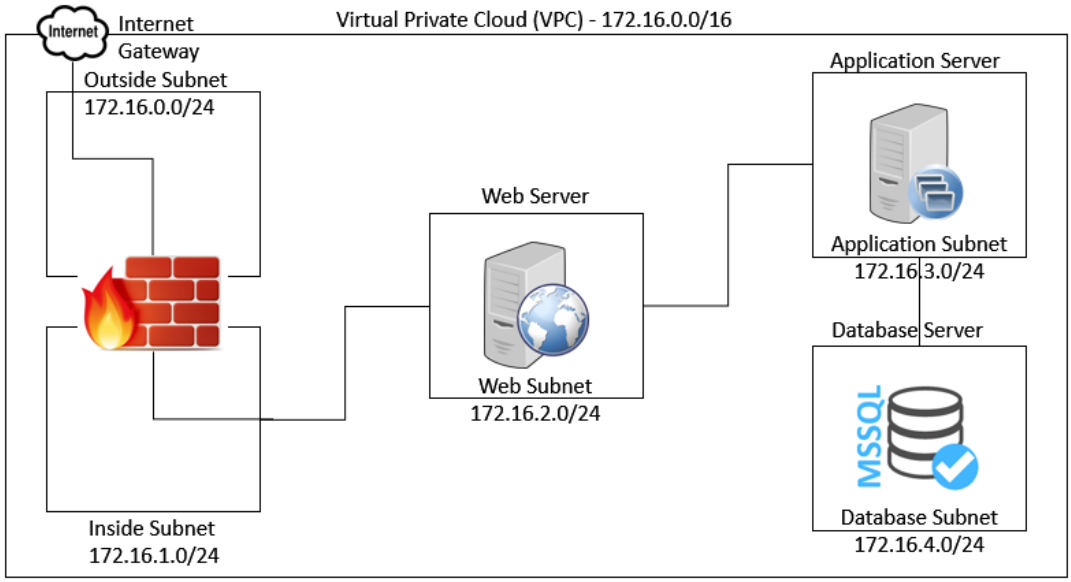


Figure 4.1 Test application deployed in AWS.

4.1.2 3-Tier Web Application

A highly data intensive web application from the Health Care domain was selected. The selected application is associated with providing health care services and is currently installed at different health care providers. These health care providers are scattered all over Australia having different numbers of end users. End users are themselves considered as elderly, in which they do not have much of a computer literacy. Hence, agents deployed from the health care service providers reach the end users and interact with this application. Due to the privacy issues associated with the health care records, the application must be installed on an intranet while scaling based on the varying user load and performance needs, which makes this an ideal application for this research. Depending on the deployment and health care service provider, number of concurrent users vary from one to 500. Hence, the capacity allocation needs to be based on the expected workload and performance.

4.1.3 Resource and Workload Combinations

Table 4.1 presents the combinations of hardware we used to carry out the experiment. The combinations are a mix from CPU cores, memory, and the number of concurrent users. CPU Cores were varied from 1, 2, 4 and 8, while the memory was varied from 2, 4, 6, and 8 for each combination of CPU Cores. This creates 16 (4 × 4) parameter combinations. We executed the user scenario for each combination that for the

concurrent users from one to 500 in different batches. Number of sessions were equal to the number of concurrent users we used to generate the load.

Table 4.1 Possible combinations for hardware and users.

CPU Cores	1, 2, 4, 8
Memory (GB)	2, 4, 6, 8
Concurrent Users	1 to 500

Having the AWS VMs from the t2 class, which is configured to ideally match with the general-purpose web applications helped us to narrow down the hardware combinations to the classes available in AWS. Table 4.2 presents the possible combination of resources we could derive based on required CPU and memory. We used the standard VMs as shown in Table 4.2. For example, T2.Micro instance type represents an instance with 1 CPU Core and 1 GB of RAM. The entire experiment was carried out using five different VM instances, namely T2.Micro, T2.Small, T2.Medium, T2.Large, and T2.ExtraLarge. Separating out JMeter load generator to a different VM made sure that it produced the same load to the application VMs in all the scenarios. Table 4.3 lists the configurations of AWS VMs and Table 4.4 shows the Amazon EC2 instance types we used for this experiment.

Table 4.2 Hardware configuration for the 3-tier sample application.

Instance Type	CPU Cores	Memory (GB)	Ramp Up	Think Time	Ramped Down
1.vmT2.Micro	1	1	10 mins	2 mins	10 mins
2.vmT2.Small	1	2	10 mins	2 mins	10 mins
3.vmT2.Medium	2	4	10 mins	2 mins	10 mins
4.vmT2.Large	2	8	10 mins	2 mins	10 mins
5.vmT2.XLarge	4	16	10 mins	2 mins	10 mins

We used the same intranet deployment topology for the selected commercial application, which gave us a setup similar to the production environment. Table 4.3 shows the configurations of the AWS VMs used for the instance type t2.large. The instance type t2 is the classification from AWS for the general-purpose web Applications. We managed to have the application deployed on different capacities by changing the instance type of the VM from t2.micro, t2.small, t2.medium, t2, large and t2.extralarge as shown in Table 4.4.

Table 4.3 Configuration of AWS VMs.

AWS VM Instance	Instance Type	Processing Power (GHz)	Memory (GB)	Disk Space (GB)
Database Server Instance	t2.large	2.4	8	70
Web Server	t2.large	2.4	8	30
Application Server	t2.large	2.4	8	30
Firewall and Proxy Server	t2.medium	2.4	3.75	100
JMeter Console + Load Generator	t2.large	2.4	8	30
JMeter Load Generator	t2.large	2.4	8	30

Table 4.4 EC2 Instances used with CPU and memory.

Instance Type	CPU Cores	Memory (GB)
1.vmT2.Micro	1	1
2.vmT2.Small	1	2
3.vmT2.Medium	2	4
4.vmT2.Large	2	8
5.vmT2.XLarge	4	16

The sample application was deployed in AWS VMs. Microsoft SQL instance was used as the database, while the application was deployed as a combination of the web server and application server.

Rubis application was also deployed in AWS. The sample commercial application and the Rubis were deployed to the Windows environment. Rubis used MySQL as the database instance. The application was deployed to the web server and application server as shown in Figure 4.2. We configured a firewall and a proxy to maintain the same deployment topology in the cloud as well. JMeter was used to generate the workload for the varying number of concurrent users for the chosen scenarios.

We selected a set of scenarios to get a wider coverage of the application functionality, as well as to hit the database as in its typical use. See Appendix A and Appendix B for the scenarios chosen for the evaluation.

Ramp Up and Ramp down time was set to 10 minutes as JMeter should get all the threads sent for the execution. Ramp up should be enough to avoid unnecessary and large workload from the beginning of the test execution. Think time was set 2 minutes

in the range of 0 to 4 with an average of 2 minutes, at the load generator to have a better variation in a random fashion. The results were collected during the steady state.

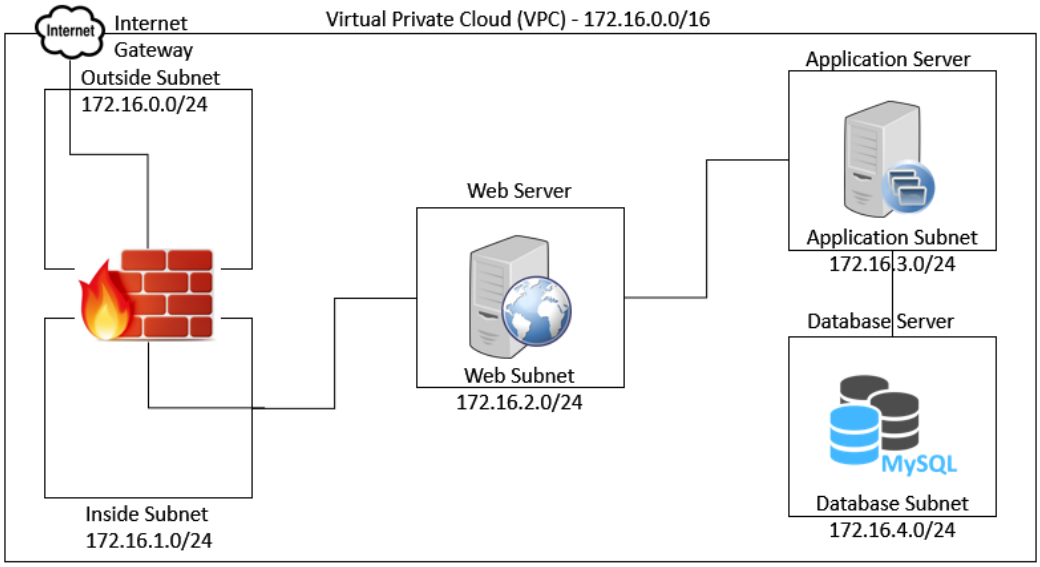


Figure 4.2 Rubis application deployed in AWS.

Both applications have a thin web front end deployed as the web server. Web server forwards the requests to the application server for further processing while handling the necessary business logic. Due to this design, application server handles a relatively higher workload, e.g., for the user scenarios and the load we applied application server introduced the highest latency to a response. This is the typical case in most 3-tier web applications where the web server mostly act as a reverse proxy, while application server does the real work. The minimum capacity requirement of commercial database servers is well known and vendors usually provide further guidelines on number of queries per second and type of queries. Therefore, the most difficulty in estimating capacity requirement lies with the application servers, which is expected to run the custom developed web application. Therefore, we consider the capacity estimation as the capacity of the application server.

Based on the nature of the chosen application, novice users must use the search feature to navigate from the point after the log in. Based on the deployment at client end, different deployments will have a different set of data volumes. In certain deployments, application caters only to a few end users with low data volume to be searched for. However, the search will take a considerable amount of time when the

deployment is for a larger user base due to large data volume. Search time will also vary based on the parameters provided and the number of database tables touched based on the parameters given to the search.

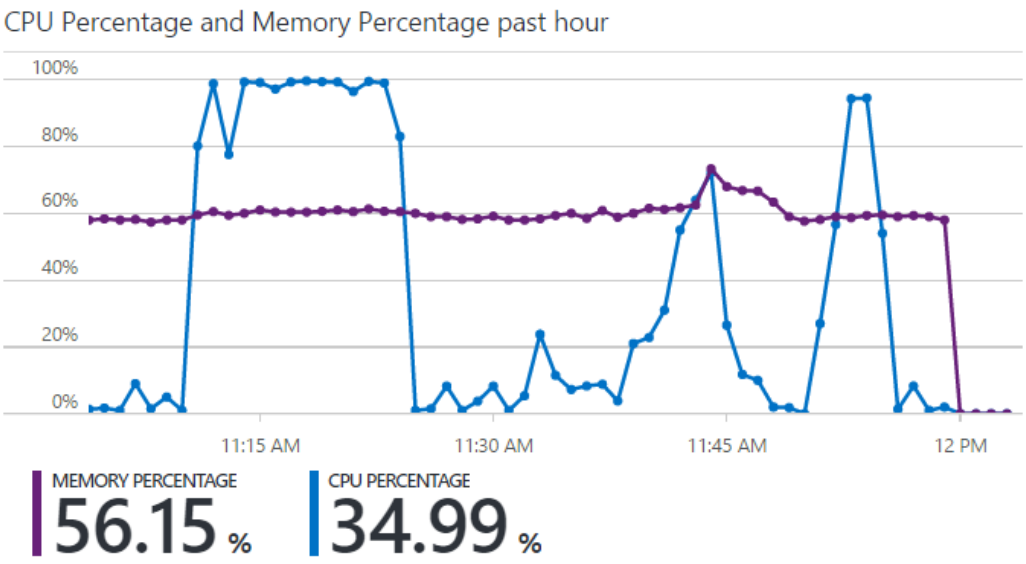
Selecting and adding a basic entity is considered as a most common step in most of the user scenarios. Selecting, editing, and saving special entities is the next major step. There are special entities with very simple structures, in which structure could be obtained from a single table, while certain special entities having complex structures in which multiple tables are required with complex joins and associations.

Logging out from the application is not only closing the application, but also involves adding logs and putting a load on the database. This is another key scenario we need focus in our experiment to have our load widely spread. Therefore, JMeter generated the load from log in to the application then executed different set of loads under different scenarios and finally the logging out from the app.
Following list shows the user scenarios selected from the 3-tier web application.

- Launching the App
- Login to the App
- Select Advanced Search
- Enter n Fields to the Search
- Select Advanced Search Enter Fields N Search
- Select a Basic Entity
- Add a Basic Entity
- Select a Special Entity A
- Edit & Save and Entity
- Save a generated Entity
- Save a Special Entity B
- Save a Special Entity C
- Navigate different sections in Special Entity D
- Edit & Save and Entity
- Select Entity & enter fields
- Save a Special Entity E
- Logout

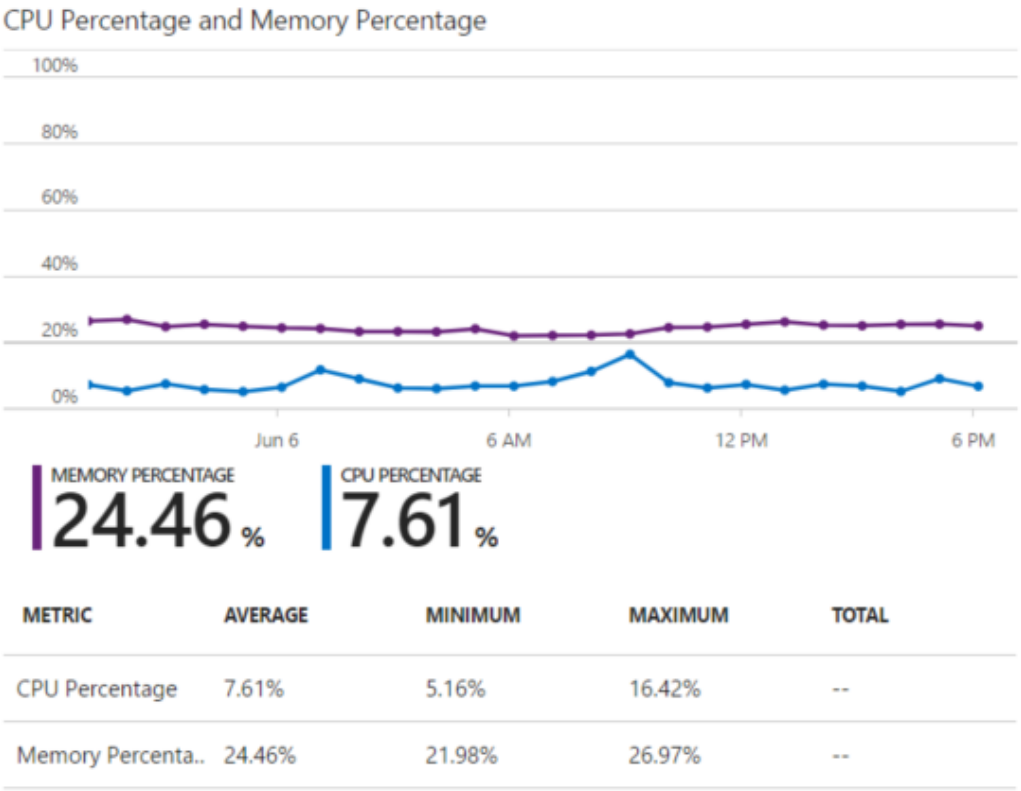
4.1.4 Data Collection

We collected and recorded the test results from the load test for the commercial application and Rubis for the user scenarios from the system we set up with JMeter. Appendix A and Appendix B shows the initial collection of test data, which shows the latency for each step against the hardware and the number of user load. We took 80% from the data collected as a subset of the train data to build the model. The remaining 20% was used to evaluate the model. The load test data was collected against the application server, as required the highest among of resources. The Figure 4.3 shows the memory and CPU utilization of the Application server and the Figure 4.4 shows the memory and CPU utilization of the Web server during the peak time from the production deployment. It clearly shows that Application server is busy processing the rendering logic to generate the JSON files, while the Web server delegates the tasks to the Application server.



METRIC	AVERAGE	MINIMUM	MAXIMUM	TOTAL
CPU Percentage	34.99%	0%	99.67%	--
Memory Percenta..	56.15%	0%	73.33%	--

Figure 4.3 Memory and CPU utilization of Application server.



METRIC	AVERAGE	MINIMUM	MAXIMUM	TOTAL
CPU Percentage	7.61%	5.16%	16.42%	--
Memory Percenta..	24.46%	21.98%	26.97%	--

Figure 4.4 Memory and CPU utilization of Web server.

4.2 Model Evaluation

There were few decisions made during the process of carrying out the experiment. Determining the proper algorithm for the evaluation model was the major decision as it affects the overall result of the experiment. In determining the ideal algorithm, selecting the right weight for the database operations was the other key to accuracy. Mapping of each applications user scenarios in a way that we have a universal approach to determine the user load is an important aspect of this whole exercise.

4.2.1 Numerical Values for Workloads

Steps fall under different categories where some involve only the Web server, some involve Web and Application servers, while others involve all three servers. When a database is involved the data touches one or more tables. Therefore, the number of tables that each step touches can be considered as a workload parameter. The nature of database operation (e.g., search vs. insert) is the other parameter.

The first experiment was carried out by assigning equal weights to the different types of operations using the multiplication of the number of database tables with the type of database operation. The numerical values calculated for each step are shown in Table 4.5 when all the database operations were treated equally. The values collected from the sample values as mentioned in Appendix C was combined with the numerical values obtained from the equal weight.

Table 4.5 Sub Scenarios with equal weight for database operations in sample application.

Steps	Database Tables	Query	Create	Update	Weight for Operation Read=1, Create=1, Update=1
Access Home Page	0	0	0	0	0
Login to the system	2	1	0	1	4
Edit configurations	4	1	0	0	4
Save Configurations	6	0	0	1	6
Perform Advanced Search	8	1	0	0	8
Select an Entity	4	1	0	0	4
Create an Entity	4	0	1	0	4
Generate an Entity	6	0	0	1	6
Navigate among the Pages	0	0	0	0	0
Logout	2	0	0	1	2

We first tried to predict the capacity using the K-Nearest Neighbour (KNN) algorithm. RStudio [23] version 1.0.143 was used to generate the evaluation model using the KNN. Different models were generated using different values for *k* starting from 1 to 10. The comparison of model predicted against the test data are shown in Figure 4.3 as follows. This prediction resulted in lots of deviation with the test data. Hence, assigning numerical values to the step based on the equal weights from database operations does not give a good evaluation model to proceed further. Even the distribution of numerical values does not show a much variation as it expands into a limited set only from 0 to 8.

As complexities of each database operation are different, we then introduced a weight or cost based on the complexity of executing a query. For example, a read database operation costs less compared to an update operation. Therefore, we set different weights such that Read = 1, Create = 2, and Update=3. Table 4.6 shows the corresponding numerical values for step when the varying weights are used.

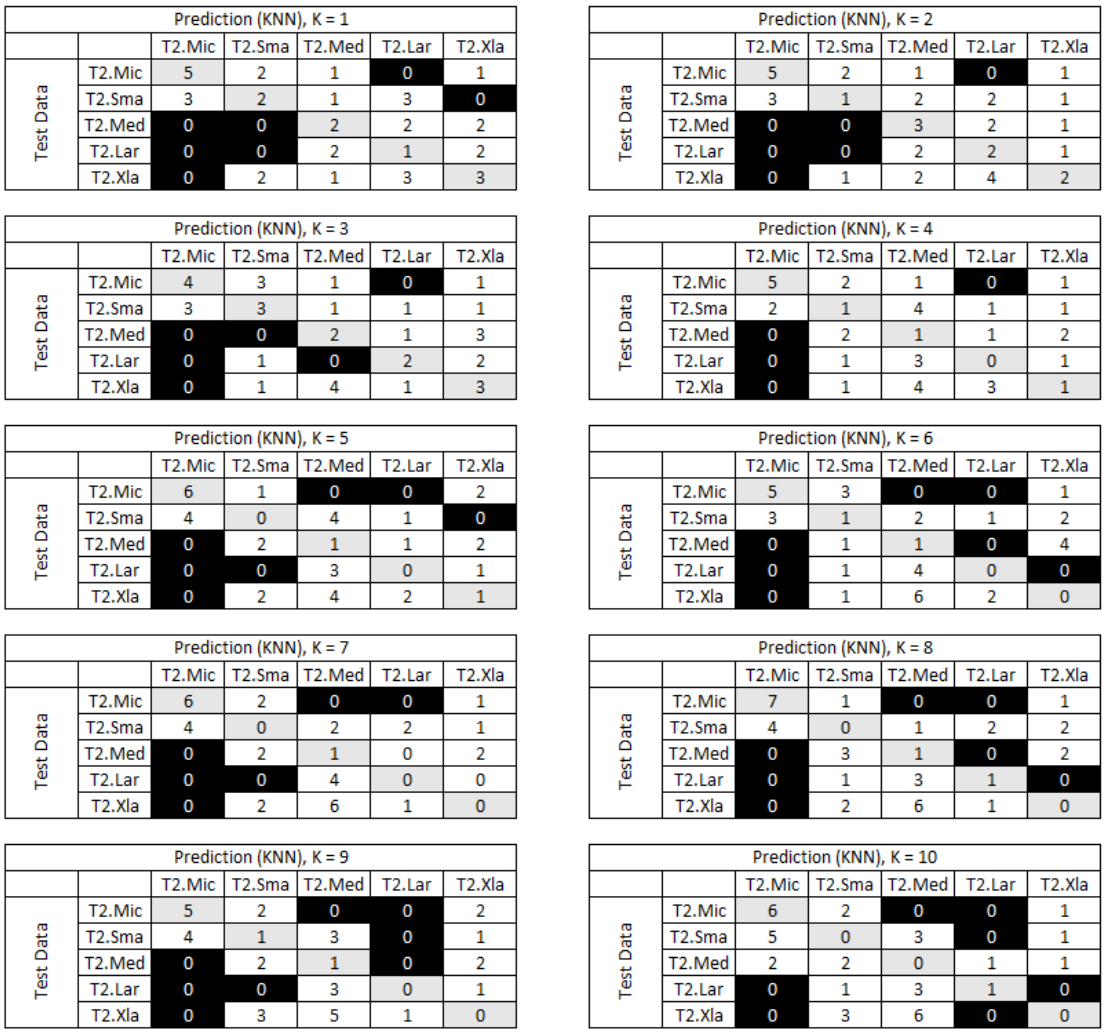


Figure 4.5 KNN algorithm on sample data with equal weights.

Table 4.6 Varying weight for database operations in sample app.

Steps	**Database Tables**	**Query**	**Create**	**Update**	**Weight for Operation Read=1, Create=2, Update=3**
Access Home Page	0	0	0	0	0
Login to the system	2	1	0	1	8
Edit configurations	4	1	0	0	4
Save Configurations	6	0	0	1	18
Perform Advanced Search	8	1	0	0	8
Select an Entity	4	1	0	0	4
Create an Entity	4	0	1	0	8
Generate an Entity	6	0	0	1	18
Navigate among the Pages	0	0	0	0	0
Logout	2	0	0	1	6

The values collected from the sample values in Appendix C was combined with the numerical values obtained from the varying weight and the model was predicted using

the KNN algorithm again. Different models were generated using different values for K starting from 1 to 10. The comparison of model predicted against the test data are shown in Figure 4.4. In this case, the predictions had less deviation with the test data. Hence, assigning numerical values to the step based on the complexity of database operations gives a better evaluation model to proceed further. Even the distribution of numerical values shows a much variation as it expands in a set only from 0 to 18.

Prediction (KNN), K = 1

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	1	0	0	0
	T2.Sma	2	3	0	3	0
	T2.Med	0	3	2	1	0
	T2.Lar	0	1	4	1	1
	T2.Xla	0	2	6	2	2

Prediction (KNN), K = 2

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	5	0	0	0	0
	T2.Sma	1	2	2	3	0
	T2.Med	0	2	3	1	0
	T2.Lar	0	1	4	1	1
	T2.Xla	0	3	4	3	2

Prediction (KNN), K = 3

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	1	0	0	0
	T2.Sma	2	3	2	1	0
	T2.Med	0	0	4	0	2
	T2.Lar	0	0	6	0	1
	T2.Xla	0	4	4	3	1

Prediction (KNN), K = 4

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	5	0	0	0	0
	T2.Sma	1	4	3	0	0
	T2.Med	0	1	3	2	0
	T2.Lar	0	0	3	4	0
	T2.Xla	0	0	2	6	4

Prediction (KNN), K = 5

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	1	0	0	0
	T2.Sma	3	1	0	3	1
	T2.Med	0	1	1	3	1
	T2.Lar	0	1	6	0	0
	T2.Xla	1	2	4	5	0

Prediction (KNN), K = 6

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	1	0	0	0
	T2.Sma	2	2	0	3	1
	T2.Med	0	1	1	3	1
	T2.Lar	0	0	6	0	1
	T2.Xla	1	3	5	3	0

Prediction (KNN), K = 7

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	0	1	0	0
	T2.Sma	1	3	1	3	0
	T2.Med	0	2	1	3	0
	T2.Lar	0	0	7	0	0
	T2.Xla	1	5	2	3	1

Prediction (KNN), K = 8

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	0	1	0	0
	T2.Sma	1	2	2	3	0
	T2.Med	0	2	1	2	1
	T2.Lar	0	0	5	0	2
	T2.Xla	1	3	4	3	1

Prediction (KNN), K = 9

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	0	1	0	0
	T2.Sma	2	1	2	2	1
	T2.Med	0	2	0	2	2
	T2.Lar	0	0	6	0	1
	T2.Xla	2	4	1	3	2

Prediction (KNN), K = 10

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	4	0	1	0	0
	T2.Sma	3	0	2	2	1
	T2.Med	0	2	1	3	0
	T2.Lar	0	0	6	0	1
	T2.Xla	0	4	4	2	2

Figure 4.6 KNN algorithm on sample data with different weights.

The decision was the varying values to the Database operations to determine the numerical value for the steps.

4.2.2 Applying More Machine Learning Models

K-Nearest neighbour is not the best model, as it does not give an evaluation model with a higher accuracy. This could be due to none linear increase in number of CPU cores and memory, which leads to an unbalanced distribution of data. Then we tried Support Vector Machine (SVM) algorithm. The values collected from the sample values (as in Appendix C) was combined with the numerical values obtained from the varying weight and the model was predicted using SVM implemented using RStudio. Different models were generated using different values for cost varying across 0.1, 1, 10, and 100. The comparison of model predicted against the test data are shown in Figure 4.5. However, this resulted in even less accuracy compared to KNN, where the overall accuracy was less than 40%. Accuracy did not improve much with varying cost values. This is probably due to a lower number of dimensions in our dataset where KNN tend to work better.

Prediction (SVM), Cost = 0.1, Accuracy = 0.3421053

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	6	0	0	0	0
	T2.Sma	4	4	3	3	3
	T2.Med	0	0	0	3	0
	T2.Lar	0	0	1	1	2
	T2.Xla	0	2	3	1	2

Prediction (SVM), Cost = 1, Accuracy = 0.2894737

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	5	4	0	0	0
	T2.Sma	4	5	3	1	2
	T2.Med	0	0	0	0	2
	T2.Lar	0	0	0	0	0
	T2.Xla	0	1	5	6	1

Prediction (SVM), Cost = 100, Accuracy = 0.3684211

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	6	3	1	0	1
	T2.Sma	1	3	3	4	2
	T2.Med	0	1	4	3	4
	T2.Lar	0	0	0	1	0
	T2.Xla	0	0	1	0	0

Prediction (SVM), Cost = 1000, Accuracy = 0.3684211

		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	7	3	0	0	1
	T2.Sma	1	3	1	2	3
	T2.Med	0	3	2	4	3
	T2.Lar	0	0	3	2	0
	T2.Xla	0	0	0	0	0

Figure 4.7 SVM algorithm on sample data with different weights.

Then we also tried the Random Forest algorithm with weighted data. The model was implemented using RStudio and different models were generated using different values for trees varying from 1 to 1,000. The comparison of model predicted against the test data are shown in Figure 4.6, where we can see good accuracy compared to test data. The best accuracy of this model was obtained when the number of trees is 100. The exact data used for this model is shown in Appendix E.

Prediction (Random Forest), Trees=1

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	5	2	0	0	1
T2.Sma	1	3	1	2	2
T2.Med	0	1	6	0	3
T2.Lar	0	1	1	3	0
T2.Xla	0	1	1	0	4

Prediction (Random Forest), Trees=10

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	6	2	0	0	0
T2.Sma	0	6	2	1	0
T2.Med	1	0	7	0	2
T2.Lar	0	0	0	5	0
T2.Xla	0	0	2	0	4

Prediction (Random Forest), Trees=20

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	6	2	0	0	0
T2.Sma	0	7	0	1	1
T2.Med	0	0	10	0	0
T2.Lar	0	2	0	2	1
T2.Xla	0	1	0	1	4

Prediction (Random Forest), Trees=30

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	8	0	0	0	0
T2.Sma	1	5	2	1	0
T2.Med	0	0	8	0	2
T2.Lar	0	0	0	3	2
T2.Xla	0	0	0	0	6

Prediction (Random Forest), Trees=50

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	8	0	0	0	0
T2.Sma	1	7	0	1	0
T2.Med	0	0	7	0	3
T2.Lar	0	1	0	3	1
T2.Xla	0	1	0	0	5

Prediction (Random Forest), Trees=100

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	8	0	0	0	0
T2.Sma	0	9	0	0	0
T2.Med	0	1	9	0	0
T2.Lar	0	0	0	3	2
T2.Xla	0	0	0	0	6

Prediction (Random Forest), Trees=200

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	7	1	0	0	0
T2.Sma	1	8	0	0	0
T2.Med	0	0	10	0	0
T2.Lar	0	0	0	4	1
T2.Xla	0	0	0	0	6

Prediction (Random Forest), Trees=500, 750, 775

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	8	0	0	0	0
T2.Sma	0	9	0	0	0
T2.Med	0	0	10	0	0
T2.Lar	0	0	0	4	1
T2.Xla	0	0	0	0	6

Prediction (Random Forest), Trees=800, 825, 1000

Test Data	T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
T2.Mic	7	1	0	0	0
T2.Sma	0	9	0	0	0
T2.Med	0	0	9	0	1
T2.Lar	0	0	0	4	1
T2.Xla	0	0	0	0	6

Figure 4.8 Random Forest algorithm on sample data with different weights.

4.3 Resource Forecasting

We looked at different machine learning algorithms to have the best possible model to predict the capacity requirement for the performance and the workload. The model with the highest accuracy was picked to move forward with this experiment. We got different models representing the relationships between capacity, performance, and resources from the training data. Table 4.7 shows the accuracy of each algorithm we used. It is clear that the model we got from the Random Forest is by far the better model to go ahead with.

As per the objective of this experiment, we used Rubis as the new application to predict the resource requirement given workload and performance target. The scenarios and steps in Rubis are different to the scenarios and steps used in our sample application. We use the same approach to calculate the numerical values for steps under each

scenarios. Table 4.8 shows the numerical values for the steps taken by assigning the varying values for different database operations.

Table 4.7 Accuracy from algorithms.

Algorithm	Accuracy
KNN (equal weight)	20.00%
KNN (varying weight)	50.00%
SVM	36.84%
Random Forest	97.14%

Table 4.8 Varying weight for Database operations in Rubis.

Steps	Database Tables	Query	Create	Update	Weight for Operation Read=1, Create=2, Update=3
Access Home Page	0	0	0	0	0
Query About me	1	1	0	0	1
Register User	1	0	1	0	2
View Item details	1	4	0	0	4
Sell Item	3	2	1	0	8
Bidding	4	3	0	2	18

The values collected from the sample values (as in Appendix A) was combined with the numerical values obtained from the varying weight and the model was predicted using the Random Forest algorithm. The exact training data given to the algorithm is in Appendix E. The difference in this evaluation is the use of Rubis data as the test data. The values collected from the Rubis application (as mentioned in Appendix B) was treated as the test data. The exact test data given to the algorithm is listed in Appendix F.

We use the model generated based on random forest and sample application training data to predict the capacity requirement for Rubis. The comparison of model prediction against the test data is shown in Figure 4.7. The accuracy of this model for the prediction against the test data is 77.42%. Therefore, the model build for one application can be used to reasonably estimate the capacity requirement of another 3-tier application.

Prediction (Random Forest), Trees = 100						
		T2.Mic	T2.Sma	T2.Med	T2.Lar	T2.Xla
Test Data	T2.Mic	3	1	1	1	1
	T2.Sma	0	5	0	0	0
	T2.Med	1	0	5	0	0
	T2.Lar	0	0	0	7	0
	T2.Xla	2	0	0	0	4

Figure 4.9 Random Forest algorithm on Rubis with different weights.

4.4 Summary

We collected a training dataset by varying number of CPU cores, memory, and the number of concurrent users. Then the steps in the dataset were mapped to a set of numerical values based on the number and type of database operations. It was realized that assigning equal weights of the database operations such as insert, update, and joins does not give an accurate model. Therefore, we assigned different weights based on the complexity of database operations. This resulted in better prediction. We tried KNN, SVM, and Random Forest algorithms to build the model, out of which Random Forest produced the most accuracy prediction of capacity. We predicted the capacity requirement for Rubis web application using the Random Forest based model and got an accuracy of 77.42% in our predictions. These results indicate that by mapping steps to a weighted numerical value a suitable machine-learning model can be used to predict the capacity requirement of a 3-tier web application given the workload and a performance target.

Chapter 5

CONCLUSIONS

Section 5.1 discuss the summary of the proposed technique and experimental results. Section 5.2 focuses on the limitations of this experiment as we scoped this work on 3-tier web applications while Section 5.3 suggests future work to extend this work to improve accuracy as well as to support applications that are more complex.

5.1 Summary

Given a workload and a performance target, we proposed a technique to predict the hardware capacity requirement. We specifically focus on 3-tier web applications, where workload is given as a set of scenarios and a corresponding set of steps as well as number of concurrent users while latency is given as the performance target. Number of CPU cores and memory is predicted as the capacity requirement.

We first collected a dataset from a sample application while varying resources (number of CPU cores and memory) and number of concurrent users against the same set of scenarios. This dataset was then used to train a set of machine learning models to capture the relationship among capacity, workload, and performance. Steps of a scenarios put different load on the servers based on the type of requests. For example, a database read is relatively simple to execute compared to an update. Therefore, we used different weights to capture the complexity of database operations. In fact, our tests results revealed that equal weights are not good at capturing the application behaviour. Among the three machine-learning algorithms we tried Random Forest was able to predict the capacity more accurately compared to K-Nearest Neighbor and Support Vector Machine algorithms. Moreover, it was able to predict the capacity for another web application with an accuracy of 77%. These results indicate that by mapping scenarios and their steps of a web application to a weighted set of numerical value a suitable machine-learning model can be used to predict the capacity requirement of a 3-tier web application given a mix of workloads and a performance target.

5.2 Limitation of Research

We specifically focused on 3-tier web applications. Even though, we were able to find a suitable model for a typical 3-tier web application, there are some limitations associated with the proposed technique.

While a 3-tier web application consists of one or more web servers, application servers, and database servers, we predict the capacity of only the application server. Hence, it is important to extend the capacity prediction to all three types of servers as a one cohesive system. While there exists a large spectrum of applications like memory intensive, CPU intensive, and bandwidth intensive, we focus on typical web applications with a set of scenarios that interacts primarily with the database. However, there are many other applications that relay on various other libraries to execute complex business logic, generate bar codes, generates PDFs, etc. Hence, those cases need more complex allocation of resources and both hardware and software level optimizations. It is a constraint that the proposed approach cannot be used to build a model for such applications.

We tested the proposed approach only against two web-based applications where one who a commercial application while Rubis was primarily used in research and training. It is important to evaluate the proposed technique across a collection of web applications.

We define the resources as number of CPU cores and memory. However, storage, bandwidth, and I/O Operations Per Second (IOPS) are other important parameters. Moreover, throughput is also an essential performance metrics. However, a larger collection of resource and performance parameters makes the number of combinations much larger when it comes to building the test dataset. Furthermore, particular language and platform used to develop a web application also have a considerable impact on resource capacity. Therefore, such factors should also be considered to be more accurate and comprehensive.

5.3 Future work

The focus of this research was to propose a new model to calculate the capacity given the performance and workload. It is interesting to extend the model to capture scenarios that are more complex. First is to predict the capacity requirement of web and database servers. Next, is to consider more complex resources such as bandwidth and IOPS. Another extension is to consider both latency and throughput as the performance metrics. Moreover, extending the proposed weights to capture none database operations such as generating a PDF file is of interest.

Even though JMeter automates load testing in each environment, it is time-consuming. Therefore, it takes lots of time to carry out the load testing over a range of VMs to collect the test dataset. Using a tool similar to Puppet [24] we can automate this process. It is suggested to test the proposed technique against other web applications and further fine-tune the model parameters.

It would be useful to expose capacity-planning as a service where the user could provide a set of application scenarios and performance target or SLA to identify the type of computing resources. This can be specifically customized for a cloud-computing environment as available types of resource configurations are known.

REFERENCES

[1] N. Mitchell and P.F. Sweeney, "On the fly capacity planning," In Proc. Object Oriented Programming Systems Languages and Applications. Oct. 2013.

[2] J. Mars, L. Tang, R. Hundt, K. Skadron, and M.L. Soffa, "Bubble-up: increasing utilization in modern warehouse scale computers via sensible co-locations," In Proc. 44th Annual IEEE/ACM Intl. Symposium on Microarchitecture, pages 248–259, 2011.

[3] D. Menascé, V. Almeida, and L. Dowdy, "Performance by Design," Prentice Hall, 2004.

[4] R. Schiesser, IT Systems Management. Prentice Hall, 2010.

[5] M. Barbacci, M.H. Klein, T.A. Longstaff, and C.B. Weinstock "Quality Attributes," [Online] Available: http://www.sei.cmu.edu/reports/95tr021.pdf.

[6] M. Careem, "Capacity Planning for Application Design Part 1," Dec. 2015.

[7] A. Nathani, S. Chaudhary, and G. Somani, "Policy based resource allocation in IaaS cloud," Future Generation Comput. Syst., vol. 28, no. 1, pp. 94-103, 2012.

[8] R.N. Calheiros, R. Ranjan, and R. Buyya. "Virtual machine provisioning based on analytical performance and Qos in cloud computing environments," In Proc. Parallel Processing, Sep. 2011.

[9] "Amazon EC2," [Online] Available: https://aws.amazon.com/ec2/

[10] "Manage Federation," [Online] Available: https://aws.amazon.com/iam/details/manage-federation/

[11] A.N. Toosi, R.N. Calheiros, R.K. Thulasiram, and R. Buyya, "Resource provisioning policies to increase IaaS provider's profit in a federated cloud environment," In Proc 13th IEEE Intl. Conf. on High Performance Computing and Communications (HPCC'11), Banff, Canada, 2011.

[12] B. Sotomayor, R.S. Montero, I.M. Llorente, and I. Foster, "An Open Source Solution for Virtual Infrastructure Management in Private and Hybrid Clouds," IEEE Internet Computing, Vol. 1, pp. 14-22, 2009.

[13] M. Asgarnezhad, R. Nasiri, and A. Shahidi, "A Systematic Method for Performance Analysis of Service Oriented Architecture Applications, Intl. Journal of Computational and Mathematical Sciences," Vol. 4, 2010.

[14] "Machine Learning Cheat Sheet," [Online] Available: http://download.microsoft.com/download/A/6/1/A613E11E-8F9C-424A-

B99D-65344785C288/microsoft-machine-learning-algorithm-cheat-sheet-v6.pdf.

[15] C.H. Papadimitriou and M. Yannakakis, "On the complexity of database queries," In Proc. 16th ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems, May 12–14, 1997, Tucson, Arizona, pp. 12–19.

[16] S. Kotsiantis, "Supervised learning: A review of classification techniques," Informatica, vol. 31, pp. 249–268, 2007.

[17] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," In L. K. Saul, Y. Weiss, and L. Bottou, editors, Advances in Neural Information Processing Systems 17, pages 513–520, Cambridge, MA, 2005. MIT Press.

[18] "Manual on setting up, using, and understanding random forests," [Online] Available: https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf

[19] S. Tong and E. Chang, "Support vector machine active learning for image retrieval," In Proc. ACM Intl. Conf. on Multimedia, pages 107-118, 2001.

[20] "Rubis," [Online] Available: http://rubis.ow2.org/

[21] "Amazon Virtual Private Cloud," [Online] Available: https://aws.amazon.com/vpc/

[22] "Apache JMeter," [Online] Available: http://jmeter.apache.org/

[23] "R Studio," [Online] Available: https://www.rstudio.com/products/rstudio/

[24] "Puppet," [Online] Available: https://puppet.com/

Appendix A: Experimental Data – Load Test Results from Sample Application

Steps	Hardware	Average latency in seconds						
		5 Users	20 Users	50 Users	75 Users	100 Users	200 Users	400 Users
Access Home Page	1.vmT2.Micro	12.032	0.887	1.4037	3.083			
Login to the system	1.vmT2.Micro	24.896	1.284	26.199	3.321			
Edit configuration	1.vmT2.Micro	5.075	3.404	10.677	7.58			
Save Configuratio ns	1.vmT2.Micro	0.3	1.914	0.569	0.997			
Perform Advanced Search	1.vmT2.Micro	0.462	0.582	0.221	0.327			
Select an Entity	1.vmT2.Micro	1.45	7.469	2.579	4.851			
Create an Entity	1.vmT2.Micro	0.088	2.764	0.894	8.448			
Generate an Entity	1.vmT2.Micro	44.366	77.34 5	131.84 3	286.38 5			
Navigate among the Pages	1.vmT2.Micro	0.288	2.565	8.231	26.686			
Logout	1.vmT2.Micro	0.461	6.218	6.358	41.672			
Access Home Page	2.vmT2.Small			1.468	2.357	18.198	10.003	
Login to the system	2.vmT2.Small			2.688	4.202	27.222	7.377	
Edit configuration	2.vmT2.Small			9.982	13.528	20.151	17.417	
Save Configuratio ns	2.vmT2.Small			10.243	12.719	0.84	1.159	
Perform Advanced Search	2.vmT2.Small			2.094	13.737	0.425	0.644	
Select an Entity	2.vmT2.Small			30.84	39.236	4.639	13.442	
Create an Entity	2.vmT2.Small			5.329	12.607	2.088	10.537	
Generate an Entity	2.vmT2.Small			141.87 7	244.87 1	326.42 9	443.21 9	
Navigate among the Pages	2.vmT2.Small			8.104	3.838	40.912	50.946	
Logout	2.vmT2.Small			28.276	20.772	45.506	82.27	
Access Home Page	3.vmT2.Mediu m				13.59	50.754	40.251	41.02 7

Steps	Hardware	Average response time in seconds						
		5 Users	20 Users	50 Users	75 Users	100 Users	200 Users	400 Users
Login to the system	3.vmT2.Medium				27.781	12.34	57.24	46.093
Edit configuration	3.vmT2.Medium				23.484	8.802	39.302	41.962
Save Configurations	3.vmT2.Medium				0.768	1.864	11.518	6.378
Perform Advanced Search	3.vmT2.Medium				0.301	1.171	5.033	3.273
Select an Entity	3.vmT2.Medium				3.961	10.826	46.888	43.123
Create an Entity	3.vmT2.Medium				1.79	1.914	21.173	55.264
Generate an Entity	3.vmT2.Medium				327.593	114.853	193.354	552.577
Navigate among the Pages	3.vmT2.Medium				8.622	0.024	16.617	25.01
Logout	3.vmT2.Medium				19.554	0.251	47.682	34.464
Access Home Page	4.vmT2.Large					3.489	18.044	68.131
Login to the system	4.vmT2.Large					3.264	40.685	116.144
Edit configuration	4.vmT2.Large					6.484	56.405	54.841
Save Configurations	4.vmT2.Large					0.754	11.012	4.674
Perform Advanced Search	4.vmT2.Large					0.556	4.514	2.397
Select an Entity	4.vmT2.Large					18.795	50.473	65.884
Create an Entity	4.vmT2.Large					6.713	12.686	4.546
Generate an Entity	4.vmT2.Large					366.958	258.551	181.292
Navigate among the Pages	4.vmT2.Large					36.194	9.228	14.085
Logout	4.vmT2.Large					48.518	36.32	40.717

Steps	Hardware	Average latency in seconds						
		5 Users	**20 Users**	**50 Users**	**75 Users**	**100 Users**	**200 Users**	**400 Users**
Access Home Page	5.vmT2.XLarge				11.989	6.481	21.66	48.636
Login to the system	5.vmT2.XLarge				20.361	8.427	50.496	13.335
Edit configuration	5.vmT2.XLarge				11.568	12.688	47.5	17.511
Save Configurations	5.vmT2.XLarge				0.595	4.154	10.604	2.659
Perform Advanced Search	5.vmT2.XLarge				0.253	1.92	2.857	1.318
Select an Entity	5.vmT2.XLarge				3.696	45.471	47.489	20.378
Create an Entity	5.vmT2.XLarge				1.657	5.055	15.043	4.784
Generate an Entity	5.vmT2.XLarge				347.439	115.358	128.684	413.918
Navigate among the Pages	5.vmT2.XLarge				9.731	19.73	10.832	36.276
Logout	5.vmT2.XLarge				26.787	61.339	46.61	86.05

Appendix B: Experimental data – Load Test Results from Rubis

Steps	Hardware	Average latency in milliseconds			
		50 Users	100 Users	200 Users	400 Users
Access Home Page	1.vmT2.Micro				16
Query About me	1.vmT2.Micro			4	
Register User	1.vmT2.Micro		5		
View Item details	1.vmT2.Micro		19		
Sell Item	1.vmT2.Micro				13
Bidding	1.vmT2.Micro			28	
Access Home Page	2.vmT2.Small			5	
Query About me	2.vmT2.Small				5
Register User	2.vmT2.Small	5			19
View Item details	2.vmT2.Small	21			
Query About me	3.vmT2.Medium	10			
Register User	3.vmT2.Medium			11	10
View Item details	3.vmT2.Medium			28	
Sell Item	3.vmT2.Medium	26			
Bidding	3.vmT2.Medium		40		
Access Home Page	4.vmT2.Large				1
Query About me	4.vmT2.Large			5	
Register User	4.vmT2.Large	14			8
View Item details	4.vmT2.Large		22		
Sell Item	4.vmT2.Large			14	
Bidding	4.vmT2.Large				31
Access Home Page	5.vmT2.XLarge	1			
Query About me	5.vmT2.XLarge				
Register User	5.vmT2.XLarge		14		13
View Item details	5.vmT2.XLarge				35
Sell Item	5.vmT2.XLarge			25	
Bidding	5.vmT2.XLarge	74			

Appendix C: Arranged data – Load Test Results from Sample Application

	Steps	Users	Latency	Hardware
1	Access Home Page	5	12.032	1.vmT2.Micro
2	Access Home Page	20	0.887	1.vmT2.Micro
3	Access Home Page	50	1.4037	1.vmT2.Micro
4	Access Home Page	75	3.083	1.vmT2.Micro
5	Login to the system	5	24.896	1.vmT2.Micro
6	Login to the system	20	1.284	1.vmT2.Micro
7	Login to the system	50	26.199	1.vmT2.Micro
8	Login to the system	75	3.321	1.vmT2.Micro
9	Edit configurations	5	5.075	1.vmT2.Micro
10	Edit configurations	20	3.404	1.vmT2.Micro
11	Edit configurations	50	10.677	1.vmT2.Micro
12	Edit configurations	75	7.58	1.vmT2.Micro
13	Save Configurations	5	0.3	1.vmT2.Micro
14	Save Configurations	20	1.914	1.vmT2.Micro
15	Save Configurations	50	0.569	1.vmT2.Micro
16	Save Configurations	75	0.997	1.vmT2.Micro
17	Perform Advanced Search	5	0.462	1.vmT2.Micro
18	Perform Advanced Search	20	0.582	1.vmT2.Micro
19	Perform Advanced Search	50	0.221	1.vmT2.Micro
20	Perform Advanced Search	75	0.327	1.vmT2.Micro
21	Select an Entity	5	1.45	1.vmT2.Micro
22	Select an Entity	20	7.469	1.vmT2.Micro
23	Select an Entity	50	2.579	1.vmT2.Micro
24	Select an Entity	75	4.851	1.vmT2.Micro
25	Create an Entity	5	0.088	1.vmT2.Micro
26	Create an Entity	20	2.764	1.vmT2.Micro
27	Create an Entity	50	0.894	1.vmT2.Micro
28	Create an Entity	75	8.448	1.vmT2.Micro
29	Generate an Entity	5	44.366	1.vmT2.Micro
30	Generate an Entity	20	77.345	1.vmT2.Micro
31	Generate an Entity	50	131.843	1.vmT2.Micro
32	Generate an Entity	75	286.385	1.vmT2.Micro
33	Navigate among the Pages	5	0.288	1.vmT2.Micro
34	Navigate among the Pages	20	2.565	1.vmT2.Micro
35	Navigate among the Pages	50	8.231	1.vmT2.Micro
36	Navigate among the Pages	75	26.686	1.vmT2.Micro
37	Logout	5	0.461	1.vmT2.Micro
38	Logout	20	6.218	1.vmT2.Micro
39	Logout	50	6.358	1.vmT2.Micro

	Steps	Users	Latency	Hardware
40	Logout	75	41.672	1.vmT2.Micro
41	Access Home Page	50	1.468	2.vmT2.Small
42	Access Home Page	75	2.357	2.vmT2.Small
43	Access Home Page	100	18.198	2.vmT2.Small
44	Access Home Page	200	10.003	2.vmT2.Small
45	Login to the system	50	2.688	2.vmT2.Small
46	Login to the system	75	4.202	2.vmT2.Small
47	Login to the system	100	27.222	2.vmT2.Small
48	Login to the system	200	7.377	2.vmT2.Small
49	Edit configurations	50	9.982	2.vmT2.Small
50	Edit configurations	75	13.528	2.vmT2.Small
51	Edit configurations	100	20.151	2.vmT2.Small
52	Edit configurations	200	17.417	2.vmT2.Small
53	Save Configurations	50	10.243	2.vmT2.Small
54	Save Configurations	75	12.719	2.vmT2.Small
55	Save Configurations	100	0.84	2.vmT2.Small
56	Save Configurations	200	1.159	2.vmT2.Small
57	Perform Advanced Search	50	2.094	2.vmT2.Small
58	Perform Advanced Search	75	13.737	2.vmT2.Small
59	Perform Advanced Search	100	0.425	2.vmT2.Small
60	Perform Advanced Search	200	0.644	2.vmT2.Small
61	Select an Entity	50	30.84	2.vmT2.Small
62	Select an Entity	75	39.236	2.vmT2.Small
63	Select an Entity	100	4.639	2.vmT2.Small
64	Select an Entity	200	13.442	2.vmT2.Small
65	Create an Entity	50	5.329	2.vmT2.Small
66	Create an Entity	75	12.607	2.vmT2.Small
67	Create an Entity	100	2.088	2.vmT2.Small
68	Create an Entity	200	10.537	2.vmT2.Small
69	Generate an Entity	50	141.877	2.vmT2.Small
70	Generate an Entity	75	244.871	2.vmT2.Small
71	Generate an Entity	100	326.429	2.vmT2.Small
72	Generate an Entity	200	443.219	2.vmT2.Small
73	Navigate among the Pages	50	8.104	2.vmT2.Small
74	Navigate among the Pages	75	3.838	2.vmT2.Small
75	Navigate among the Pages	100	40.912	2.vmT2.Small
76	Navigate among the Pages	200	50.946	2.vmT2.Small
77	Logout	50	28.276	2.vmT2.Small
78	Logout	75	20.772	2.vmT2.Small
79	Logout	100	45.506	2.vmT2.Small
80	Logout	200	82.27	2.vmT2.Small
81	Access Home Page	75	13.59	3.vmT2.Medium
82	Access Home Page	100	50.754	3.vmT2.Medium

	Steps	Users	Latency	Hardware
83	Access Home Page	200	40.251	3.vmT2.Medium
84	Access Home Page	400	41.027	3.vmT2.Medium
85	Login to the system	75	27.781	3.vmT2.Medium
86	Login to the system	100	12.34	3.vmT2.Medium
87	Login to the system	200	57.24	3.vmT2.Medium
88	Login to the system	400	46.093	3.vmT2.Medium
89	Edit configurations	75	23.484	3.vmT2.Medium
90	Edit configurations	100	8.802	3.vmT2.Medium
91	Edit configurations	200	39.302	3.vmT2.Medium
92	Edit configurations	400	41.962	3.vmT2.Medium
93	Save Configurations	75	0.768	3.vmT2.Medium
94	Save Configurations	100	1.864	3.vmT2.Medium
95	Save Configurations	200	11.518	3.vmT2.Medium
96	Save Configurations	400	6.378	3.vmT2.Medium
97	Perform Advanced Search	75	0.301	3.vmT2.Medium
98	Perform Advanced Search	100	1.171	3.vmT2.Medium
99	Perform Advanced Search	200	5.033	3.vmT2.Medium
100	Perform Advanced Search	400	3.273	3.vmT2.Medium
101	Select an Entity	75	3.961	3.vmT2.Medium
102	Select an Entity	100	10.826	3.vmT2.Medium
103	Select an Entity	200	46.888	3.vmT2.Medium
104	Select an Entity	400	43.123	3.vmT2.Medium
105	Create an Entity	75	1.79	3.vmT2.Medium
106	Create an Entity	100	1.914	3.vmT2.Medium
107	Create an Entity	200	21.173	3.vmT2.Medium
108	Create an Entity	400	55.264	3.vmT2.Medium
109	Generate an Entity	75	327.593	3.vmT2.Medium
110	Generate an Entity	100	114.853	3.vmT2.Medium
111	Generate an Entity	200	193.354	3.vmT2.Medium
112	Generate an Entity	400	552.577	3.vmT2.Medium
113	Navigate among the Pages	75	8.622	3.vmT2.Medium
114	Navigate among the Pages	100	0.024	3.vmT2.Medium
115	Navigate among the Pages	200	16.617	3.vmT2.Medium
116	Navigate among the Pages	400	25.01	3.vmT2.Medium
117	Logout	75	19.554	3.vmT2.Medium
118	Logout	100	0.251	3.vmT2.Medium
119	Logout	200	47.682	3.vmT2.Medium
120	Logout	400	34.464	3.vmT2.Medium
121	Access Home Page	100	3.489	4.vmT2.Large
122	Access Home Page	200	18.044	4.vmT2.Large
123	Access Home Page	400	68.131	4.vmT2.Large
124	Login to the system	100	3.264	4.vmT2.Large
125	Login to the system	200	40.685	4.vmT2.Large

	Steps	Users	Latency	Hardware
126	Login to the system	400	116.144	4.vmT2.Large
127	Edit configurations	100	6.484	4.vmT2.Large
128	Edit configurations	200	56.405	4.vmT2.Large
129	Edit configurations	400	54.841	4.vmT2.Large
130	Save Configurations	100	0.754	4.vmT2.Large
131	Save Configurations	200	11.012	4.vmT2.Large
132	Save Configurations	400	4.674	4.vmT2.Large
133	Perform Advanced Search	100	0.556	4.vmT2.Large
134	Perform Advanced Search	200	4.514	4.vmT2.Large
135	Perform Advanced Search	400	2.397	4.vmT2.Large
136	Select an Entity	100	18.795	4.vmT2.Large
137	Select an Entity	200	50.473	4.vmT2.Large
138	Select an Entity	400	65.884	4.vmT2.Large
139	Create an Entity	100	6.713	4.vmT2.Large
140	Create an Entity	200	12.686	4.vmT2.Large
141	Create an Entity	400	4.546	4.vmT2.Large
142	Generate an Entity	100	366.958	4.vmT2.Large
143	Generate an Entity	200	258.551	4.vmT2.Large
144	Generate an Entity	400	181.292	4.vmT2.Large
145	Navigate among the Pages	100	36.194	4.vmT2.Large
146	Navigate among the Pages	200	9.228	4.vmT2.Large
147	Navigate among the Pages	400	14.085	4.vmT2.Large
148	Logout	100	48.518	4.vmT2.Large
149	Logout	200	36.32	4.vmT2.Large
150	Logout	400	40.717	4.vmT2.Large
151	Access Home Page	75	11.989	5.vmT2.XLarge
152	Access Home Page	100	6.481	5.vmT2.XLarge
153	Access Home Page	200	21.66	5.vmT2.XLarge
154	Access Home Page	400	48.636	5.vmT2.XLarge
155	Login to the system	75	20.361	5.vmT2.XLarge
156	Login to the system	100	8.427	5.vmT2.XLarge
157	Login to the system	200	50.496	5.vmT2.XLarge
158	Login to the system	400	13.335	5.vmT2.XLarge
159	Edit configurations	75	11.568	5.vmT2.XLarge
160	Edit configurations	100	12.688	5.vmT2.XLarge
161	Edit configurations	200	47.5	5.vmT2.XLarge
162	Edit configurations	400	17.511	5.vmT2.XLarge
163	Save Configurations	75	0.595	5.vmT2.XLarge
164	Save Configurations	100	4.154	5.vmT2.XLarge
165	Save Configurations	200	10.604	5.vmT2.XLarge
166	Save Configurations	400	2.659	5.vmT2.XLarge
167	Perform Advanced Search	75	0.253	5.vmT2.XLarge
168	Perform Advanced Search	100	1.92	5.vmT2.XLarge

	Steps	Users	Latency	Hardware
169	Perform Advanced Search	200	2.857	5.vmT2.XLarge
170	Perform Advanced Search	400	1.318	5.vmT2.XLarge
171	Select an Entity	75	3.696	5.vmT2.XLarge
172	Select an Entity	100	45.471	5.vmT2.XLarge
173	Select an Entity	200	47.489	5.vmT2.XLarge
174	Select an Entity	400	20.378	5.vmT2.XLarge
175	Create an Entity	75	1.657	5.vmT2.XLarge
176	Create an Entity	100	5.055	5.vmT2.XLarge
177	Create an Entity	200	15.043	5.vmT2.XLarge
178	Create an Entity	400	4.784	5.vmT2.XLarge
179	Generate an Entity	75	347.439	5.vmT2.XLarge
180	Generate an Entity	100	115.358	5.vmT2.XLarge
181	Generate an Entity	200	128.684	5.vmT2.XLarge
182	Generate an Entity	400	413.918	5.vmT2.XLarge
183	Navigate among the Pages	75	9.731	5.vmT2.XLarge
184	Navigate among the Pages	100	19.73	5.vmT2.XLarge
185	Navigate among the Pages	200	10.832	5.vmT2.XLarge
186	Navigate among the Pages	400	36.276	5.vmT2.XLarge
187	Logout	75	26.787	5.vmT2.XLarge
188	Logout	100	61.339	5.vmT2.XLarge
189	Logout	200	46.61	5.vmT2.XLarge
190	Logout	400	86.05	5.vmT2.XLarge

Appendix D: Arranged data – Load Test Results from Rubis

	Steps	Users	Latency	Hardware
1	Register User	400	16	1.vmT2.Micro
2	Access Home Page	400	16	1.vmT2.Micro
3	Query About me	200	4	1.vmT2.Micro
4	Register User	100	5	1.vmT2.Micro
5	View Item details	100	19	1.vmT2.Micro
6	Sell Item	400	13	1.vmT2.Micro
7	Bidding	200	28	1.vmT2.Micro
8	Register User	400	19	2.vmT2.Small
9	Access Home Page	200	5	2.vmT2.Small
10	Query About me	400	5	2.vmT2.Small
11	Register User	100	5	2.vmT2.Small
12	View Item details	50	21	2.vmT2.Small
13	Register User	400	10	3.vmT2.Medium
14	Query About me	50	10	3.vmT2.Medium
15	Register User	200	11	3.vmT2.Medium
16	View Item details	200	28	3.vmT2.Medium
17	Sell Item	50	26	3.vmT2.Medium
18	Bidding	100	40	3.vmT2.Medium
19	Register User	400	8	4.vmT2.Large
20	Access Home Page	400	1	4.vmT2.Large
21	Query About me	200	5	4.vmT2.Large
22	Register User	50	14	4.vmT2.Large
23	View Item details	100	22	4.vmT2.Large
24	Sell Item	200	14	4.vmT2.Large
25	Bidding	400	31	4.vmT2.Large
26	Register User	400	13	5.vmT2.XLarge
27	Access Home Page	50	1	5.vmT2.XLarge
28	Register User	100	14	5.vmT2.XLarge
29	View Item details	400	35	5.vmT2.XLarge
30	Sell Item	200	25	5.vmT2.XLarge
31	Bidding	50	74	5.vmT2.XLarge

Appendix E: Processed data – Load Test Results from Sample Application

	Steps	Users	Latency	Hardware
1	0	5	12.032	1.vmT2.Micro
2	0	20	0.887	1.vmT2.Micro
3	0	50	1.4037	1.vmT2.Micro
4	0	75	3.083	1.vmT2.Micro
5	8	5	24.896	1.vmT2.Micro
6	8	20	1.284	1.vmT2.Micro
7	8	50	26.199	1.vmT2.Micro
8	8	75	3.321	1.vmT2.Micro
9	4	5	5.075	1.vmT2.Micro
10	4	20	3.404	1.vmT2.Micro
11	4	50	10.677	1.vmT2.Micro
12	4	75	7.58	1.vmT2.Micro
13	18	5	0.3	1.vmT2.Micro
14	18	20	1.914	1.vmT2.Micro
15	18	50	0.569	1.vmT2.Micro
16	18	75	0.997	1.vmT2.Micro
17	8	5	0.462	1.vmT2.Micro
18	8	20	0.582	1.vmT2.Micro
19	8	50	0.221	1.vmT2.Micro
20	8	75	0.327	1.vmT2.Micro
21	4	5	1.45	1.vmT2.Micro
22	4	20	7.469	1.vmT2.Micro
23	4	50	2.579	1.vmT2.Micro
24	4	75	4.851	1.vmT2.Micro
25	8	5	0.088	1.vmT2.Micro
26	8	20	2.764	1.vmT2.Micro
27	8	50	0.894	1.vmT2.Micro
28	8	75	8.448	1.vmT2.Micro
29	18	5	44.366	1.vmT2.Micro
30	18	20	77.345	1.vmT2.Micro
31	18	50	131.843	1.vmT2.Micro
32	18	75	286.385	1.vmT2.Micro
33	0	5	0.288	1.vmT2.Micro
34	0	20	2.565	1.vmT2.Micro
35	0	50	8.231	1.vmT2.Micro
36	0	75	26.686	1.vmT2.Micro
37	6	5	0.461	1.vmT2.Micro
38	6	20	6.218	1.vmT2.Micro
39	6	50	6.358	1.vmT2.Micro

	Steps	Users	Latency	Hardware
40	6	75	41.672	1.vmT2.Micro
41	0	50	1.468	2.vmT2.Small
42	0	75	2.357	2.vmT2.Small
43	0	100	18.198	2.vmT2.Small
44	0	200	10.003	2.vmT2.Small
45	8	50	2.688	2.vmT2.Small
46	8	75	4.202	2.vmT2.Small
47	8	100	27.222	2.vmT2.Small
48	8	200	7.377	2.vmT2.Small
49	4	50	9.982	2.vmT2.Small
50	4	75	13.528	2.vmT2.Small
51	4	100	20.151	2.vmT2.Small
52	4	200	17.417	2.vmT2.Small
53	18	50	10.243	2.vmT2.Small
54	18	75	12.719	2.vmT2.Small
55	18	100	0.84	2.vmT2.Small
56	18	200	1.159	2.vmT2.Small
57	8	50	2.094	2.vmT2.Small
58	8	75	13.737	2.vmT2.Small
59	8	100	0.425	2.vmT2.Small
60	8	200	0.644	2.vmT2.Small
61	4	50	30.84	2.vmT2.Small
62	4	75	39.236	2.vmT2.Small
63	4	100	4.639	2.vmT2.Small
64	4	200	13.442	2.vmT2.Small
65	8	50	5.329	2.vmT2.Small
66	8	75	12.607	2.vmT2.Small
67	8	100	2.088	2.vmT2.Small
68	8	200	10.537	2.vmT2.Small
69	18	50	141.877	2.vmT2.Small
70	18	75	244.871	2.vmT2.Small
71	18	100	326.429	2.vmT2.Small
72	18	200	443.219	2.vmT2.Small
73	0	50	8.104	2.vmT2.Small
74	0	75	3.838	2.vmT2.Small
75	0	100	40.912	2.vmT2.Small
76	0	200	50.946	2.vmT2.Small
77	6	50	28.276	2.vmT2.Small
78	6	75	20.772	2.vmT2.Small
79	6	100	45.506	2.vmT2.Small
80	6	200	82.27	2.vmT2.Small
81	0	75	13.59	3.vmT2.Medium
82	0	100	50.754	3.vmT2.Medium

	Steps	Users	Latency	Hardware
83	0	200	40.251	3.vmT2.Medium
84	0	400	41.027	3.vmT2.Medium
85	8	75	27.781	3.vmT2.Medium
86	8	100	12.34	3.vmT2.Medium
87	8	200	57.24	3.vmT2.Medium
88	8	400	46.093	3.vmT2.Medium
89	4	75	23.484	3.vmT2.Medium
90	4	100	8.802	3.vmT2.Medium
91	4	200	39.302	3.vmT2.Medium
92	4	400	41.962	3.vmT2.Medium
93	18	75	0.768	3.vmT2.Medium
94	18	100	1.864	3.vmT2.Medium
95	18	200	11.518	3.vmT2.Medium
96	18	400	6.378	3.vmT2.Medium
97	8	75	0.301	3.vmT2.Medium
98	8	100	1.171	3.vmT2.Medium
99	8	200	5.033	3.vmT2.Medium
100	8	400	3.273	3.vmT2.Medium
101	4	75	3.961	3.vmT2.Medium
102	4	100	10.826	3.vmT2.Medium
103	4	200	46.888	3.vmT2.Medium
104	4	400	43.123	3.vmT2.Medium
105	8	75	1.79	3.vmT2.Medium
106	8	100	1.914	3.vmT2.Medium
107	8	200	21.173	3.vmT2.Medium
108	8	400	55.264	3.vmT2.Medium
109	18	75	327.593	3.vmT2.Medium
110	18	100	114.853	3.vmT2.Medium
111	18	200	193.354	3.vmT2.Medium
112	18	400	552.577	3.vmT2.Medium
113	0	75	8.622	3.vmT2.Medium
114	0	100	0.024	3.vmT2.Medium
115	0	200	16.617	3.vmT2.Medium
116	0	400	25.01	3.vmT2.Medium
117	6	75	19.554	3.vmT2.Medium
118	6	100	0.251	3.vmT2.Medium
119	6	200	47.682	3.vmT2.Medium
120	6	400	34.464	3.vmT2.Medium
121	0	100	3.489	4.vmT2.Large
122	0	200	18.044	4.vmT2.Large
123	0	400	68.131	4.vmT2.Large
124	8	100	3.264	4.vmT2.Large
125	8	200	40.685	4.vmT2.Large

	Steps	Users	Latency	Hardware
126	8	400	116.144	4.vmT2.Large
127	4	100	6.484	4.vmT2.Large
128	4	200	56.405	4.vmT2.Large
129	4	400	54.841	4.vmT2.Large
130	18	100	0.754	4.vmT2.Large
131	18	200	11.012	4.vmT2.Large
132	18	400	4.674	4.vmT2.Large
133	8	100	0.556	4.vmT2.Large
134	8	200	4.514	4.vmT2.Large
135	8	400	2.397	4.vmT2.Large
136	4	100	18.795	4.vmT2.Large
137	4	200	50.473	4.vmT2.Large
138	4	400	65.884	4.vmT2.Large
139	8	100	6.713	4.vmT2.Large
140	8	200	12.686	4.vmT2.Large
141	8	400	4.546	4.vmT2.Large
142	18	100	366.958	4.vmT2.Large
143	18	200	258.551	4.vmT2.Large
144	18	400	181.292	4.vmT2.Large
145	0	100	36.194	4.vmT2.Large
146	0	200	9.228	4.vmT2.Large
147	0	400	14.085	4.vmT2.Large
148	6	100	48.518	4.vmT2.Large
149	6	200	36.32	4.vmT2.Large
150	6	400	40.717	4.vmT2.Large
151	0	75	11.989	5.vmT2.XLarge
152	0	100	6.481	5.vmT2.XLarge
153	0	200	21.66	5.vmT2.XLarge
154	0	400	48.636	5.vmT2.XLarge
155	8	75	20.361	5.vmT2.XLarge
156	8	100	8.427	5.vmT2.XLarge
157	8	200	50.496	5.vmT2.XLarge
158	8	400	13.335	5.vmT2.XLarge
159	4	75	11.568	5.vmT2.XLarge
160	4	100	12.688	5.vmT2.XLarge
161	4	200	47.5	5.vmT2.XLarge
162	4	400	17.511	5.vmT2.XLarge
163	18	75	0.595	5.vmT2.XLarge
164	18	100	4.154	5.vmT2.XLarge
165	18	200	10.604	5.vmT2.XLarge
166	18	400	2.659	5.vmT2.XLarge
167	8	75	0.253	5.vmT2.XLarge
168	8	100	1.92	5.vmT2.XLarge

	Steps	Users	Latency	Hardware
169	8	200	2.857	5.vmT2.XLarge
170	8	400	1.318	5.vmT2.XLarge
171	4	75	3.696	5.vmT2.XLarge
172	4	100	45.471	5.vmT2.XLarge
173	4	200	47.489	5.vmT2.XLarge
174	4	400	20.378	5.vmT2.XLarge
175	8	75	1.657	5.vmT2.XLarge
176	8	100	5.055	5.vmT2.XLarge
177	8	200	15.043	5.vmT2.XLarge
178	8	400	4.784	5.vmT2.XLarge
179	18	75	347.439	5.vmT2.XLarge
180	18	100	115.358	5.vmT2.XLarge
181	18	200	128.684	5.vmT2.XLarge
182	18	400	413.918	5.vmT2.XLarge
183	0	75	9.731	5.vmT2.XLarge
184	0	100	19.73	5.vmT2.XLarge
185	0	200	10.832	5.vmT2.XLarge
186	0	400	36.276	5.vmT2.XLarge
187	6	75	26.787	5.vmT2.XLarge
188	6	100	61.339	5.vmT2.XLarge
189	6	200	46.61	5.vmT2.XLarge
190	6	400	86.05	5.vmT2.XLarge

Appendix F: Processed data – Load Test Results from Sample Application

	Steps	Users	Latency	Hardware
1	2	400	16	1.vmT2.Micro
2	0	400	16	1.vmT2.Micro
3	1	200	4	1.vmT2.Micro
4	2	100	5	1.vmT2.Micro
5	4	100	19	1.vmT2.Micro
6	8	400	13	1.vmT2.Micro
7	18	200	28	1.vmT2.Micro
8	2	400	19	2.vmT2.Small
9	0	200	5	2.vmT2.Small
10	1	400	5	2.vmT2.Small
11	2	100	5	2.vmT2.Small
12	4	50	21	2.vmT2.Small
13	2	400	10	3.vmT2.Medium
14	1	50	10	3.vmT2.Medium
15	2	200	11	3.vmT2.Medium
16	4	200	28	3.vmT2.Medium
17	8	50	26	3.vmT2.Medium
18	18	100	40	3.vmT2.Medium
19	2	400	8	4.vmT2.Large
20	0	400	1	4.vmT2.Large
21	1	200	5	4.vmT2.Large
22	2	50	14	4.vmT2.Large
23	4	100	22	4.vmT2.Large
24	8	200	14	4.vmT2.Large
25	18	400	31	4.vmT2.Large
26	2	400	13	5.vmT2.XLarge
27	0	50	1	5.vmT2.XLarge
28	2	100	14	5.vmT2.XLarge
29	4	400	35	5.vmT2.XLarge
30	8	200	25	5.vmT2.XLarge
31	18	50	74	5.vmT2.XLarge