Auto-scaling of Cloud Resources to Manage Surges in Web Application Traffic

By

Farzana Younas

NUST201260808MSEECS61312F

Supervisor:

Dr. Peter Bloodsworth

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan

August 2015

Auto-scaling of Cloud Resources to Manage Surges in Web Application Traffic

By

Farzana Younas

NUST201260808MSEECS61312F

Supervisor:

Dr. Peter Bloodsworth

Department of Computing

A thesis submitted in partial fulfillment of the requirement for the degree of Masters in Computer Science (MS CS)

In

School of Electrical Engineering and Computer Science

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

July 2015

NUST School of Electrical Engineering and Computer Sciences

A center of excellence for quality education and research

Approval

Certified that the contents of thesis document titled "Auto-scaling of Cloud Resources to Manage Surges in Web Application Traffic" submitted by Mr./Miss Farzana Younas have been found satisfactory for the requirement of degree.

Advisor: Dr. Peter Bloodsworth

Committee Member1: Dr. Hamid Mukhtar

Committee Member2: Dr. Sohail Iqbal

Committee Member3: Dr. Tahir Azim

Dedication

Dedicated to my Parents

For their profound love, support and encouragement

And my supervisor Dr. Peter Bloodsworth

For his continuous support, encouragement and guidance

CERTIFICATE OF ORIGINALITY

I hereby declare that the thesis titled "**Auto-scaling of Cloud Resources to Manage Surges in Web Application Traffic**" is my own work and to the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NIIT or any other education institute, except where due acknowledgment, is made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic is acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Farzana Younas

Signature: _____

Acknowledgement

First of all I would pay my gratitude to **ALMIGHTY ALLAH** the most Merciful and the most Beneficent. HE gave me health, strength and passion to carry this research work. After this I would like to pay my regards to my **parents** for their support and encouragement and prayers. I am sincerely grateful to my advisor **Dr. Peter Bloodsworth** for his continuous encouragement, guidance and precious time he dedicated to my research work. He has been a remarkable mentor and I am very thankful to him for his encouraging support throughout the research work.

I would like to say a big thanks to **Ghalib Ahmed Tahir** for all his assistance, suggestions and continuous motivation. I am also thankful to my GEC members **Dr. Hamid Mukhtar, Dr. Sohail Iqbal** and **Dr. Tahir Azim** for being so kind and available whenever I needed their help.

I am also grateful to all my friends and co-works for being so supportive and source of motivation for me to completing research work.

Farzana Younas

Table of Contents

1. Introduction 1
   1.1 Cloud Computing 1
   1.2 Motivation 2
   1.3 Problem Statement 3
   1.4 Methodology for Carrying the Research 4
      1.4.1 Hypothesis 4
      1.4.2 Research Questions 4
   1.5 Objectives of Research 5
   1.6 Research Contribution 6
   1.7 Thesis Organization 6
2. Literature Review 8
   2.1 Auto-Scaling in Clouds-State of the art 8
      2.1.1 Static load balancing 9
      2.1.2 Dynamic resource allocation 9
   2.2 Threshold Rule Based Techniques 10
   2.3 Predictive resource allocation techniques 12
      2.3.1 Resource usage predictions 12
      2.3.2 Workload predictions 13
   2.4 Surge Detection in Web application Traffic 14
   2.5 Conclusion 15
3. Design and Architecture 17
   3.1 Multi-agent System Architecture 17
   3.2 Monitoring Agent 19
   3.3 Data Access Agent 19
   3.4 Prediction Agent 19
   3.5 Surge Detection Agent 21
      3.5.1 Small Surge 23
      3.5.2 Large Surge 23
      3.5.3 Medium Surge 23

3.6 Auto-scaling Agent 23
3.6.1 Up Scaling of Resources 25
3.6.2 Down Scaling of Resources 27
3.6.3 Instance Buffer 27
3.7 Data Integration Agent 30
3.8 VM Management Agent 30
3.9 User Interface 30
4. Results and Discussion 33
4.1 Data and Environment Setup 33
4.2 Workload Predictions 34
4.2.1 Workload Predictions for Small Sized Surge 35
4.2.2 Workload Predictions for Large Sized Surge 37
4.2.2.1 Workload Predictions during a Large Surge 38
4.2.3 Workload Predictions for a Medium Surge 39
4.3 Surge Detection 42
4.4 Auto-Scaling 43
4.5 Conclusion 46
5. Conclusion and Future Work 49
5.1 Conclusion 49
5.2 Future work 50
References 51

List of Figures

Figure 1 Steps to Carry Research 4
Figure 2 Architecture of Multi-Agent System 17
Figure 3 Flow diagram of Auto-scaling agent 25
Figure 4 Flow Diagram for Down Scaling of Instances 28
Figure 5 Wikipedia Workload 34
Figure 6 Comparison of Actual and Predicted number of requests by Machine Learning Algorithms for Small Surge 35
Figure 7 Comparison of Actual and Predicted number of requests by Machine Learning Algorithms for Large Surge 37
Figure 8 Comparison of Actual and Predicted number of request during a surge 39
Figure 9 Comparison of actual and predicted number of requests medium surge 40
Figure 10: Total number of Requests during Time Window of Ten Minutes 41
Figure 11 Overview of total number of requests and total capacity of running instance at the moment 44
Figure 12 Auto-scaling of Instances to Adjust predicted load 45

List of Tables

Table 1 T2 instance types and their specifications 26
Table 2 CPU Credits of T2 instances..... 27
Table 3 Summary of Accuracy Measures of prediction algorithms for small surge 36
Table 4 Summary of Accuracy Measures of prediction algorithms for large surge..... 38
Table 5 Summary of Accuracy Measures of prediction algorithms during surge 39
Table 6 Summary of Accuracy Measures of prediction algorithms for medium surge.... 40
Table 7 Threshold values for Various Instance Parameters..... 42
Table 8 Alarms generated by Surge Detector 43
Table 9 Total number of requests and instances for each time window 46

ABSTRACT

Cloud computing has become a powerful computing platform for web applications because of the ability to auto-scale based on the demand from the users. Many cloud service providers including Amazon, Azure etc. and software such as RightScale and Scalr provide auto-scaling mechanisms. The current auto-scaling mechanisms seem promising when the application workload follows a certain pattern, but they do not perform well when a sudden surge in traffic hits the web application. They lack the intelligence to learn about the web applications they host. The result of this is often unavailability of the application or very poor performance when underlying application experiences an unpredictable sudden surge. Most of the existing techniques of auto-scaling are based on simple user defined metrics and resource utilization thresholds. These approaches mainly focus on the static allocation and do not work well with the present dynamic natured web applications where work load is highly unpredictable. This research work proposes a solution for autonomous dynamic scaling and reconfiguring clouds when web applications suffer a sudden large and unpredictable swing in traffic. It makes use of the intelligence of the multi-agents to detect the application behavior for unusual traffic. Machine learning techniques are applied to accurately predict the future workload and a surge detection mechanism to look for potential surges. The proposed system also contains a planner agent which carefully computes the resource requirement to handle the incoming traffic surge. We have proposed the use of a resource buffer as quick solution for the sudden surges in traffic which reduces the VM's churn time and makes sure timely availability of the resources when resource demand increases suddenly due to the surge in traffic.

CHAPTER I

1. Introduction

1.1 Cloud Computing

Large swings in the traffic of web applications are common with the increased popularity of internet. Web applications experience multiple patterns of dynamic load received from end users. A news channel may face a large spike in viewership while covering some Royal event in England. Similarly consider the example of an online ticket booking Web applications that announces discounts during Olympics and the drastic load it will experience at multiple times in multiple regions. Such sudden and dynamic changes in web traffic are difficult to manage that is why such swings mostly result in decreased performance and sometimes crash of hosting servers. This demands for a hosting environment where the capacity of the underlying system can be changed according to the needs in traffic to prevent under and over utilization of resources. Cloud infrastructure provides such a flexible environment where capacity can be increased or decreased within a very short time period to handle the varying workload. Capacity can be scaled up by adding more servers to the underlying infrastructure when workload in increased and it can be scaled down by removing servers according to the decreasing workload. The users will pay only for the resources they use. There are a number of cloud providers that offer such virtualized environment to the businesses and organizations. The clients can rent resources to meet their application demand mostly on hourly basis. However rescaling and reconfiguration of the hosting platform to handle the dynamic load of web application need a certain level of intelligence as managing and making choices on run-time is a complex task.

Cloud providers like Azure and Amazon enable system administrators to choose a wide range of arrangements regarding the configurations, size and location of virtual machines. Generally they also provide many models for rescaling based on the user demand. Their models are mostly based on raw measures such as CPU utilization, memory utilization, and number of sessions. Every web application is unique in its usage pattern and in the same

manner their resource requirements are also different for same parameters. Such profiling information can be very useful when making a decision of rescaling but current load balancers ignore such factors while distributing the load. Most vendors including Amazon look for certain parameters such as CPU utilization to cross a specified threshold point to upscale or down scale the infrastructure. These parameters show very large variability and poor auto-correlation which makes it difficult to represent large spikes accurately. Current scaling mechanisms work well when the load is known in advance which mostly not the case is. In real-world with the increasing trend of social media and e-commerce surges can occur at any time without much of a prior indication. When a traffic spike happens, there is a delay before measures can be applied in real-time because of the startup time resources take to launch. Prediction of spikes can be very useful to achieve elasticity. Our research explores the potential for predicting spikes sufficiently in advance so that preventive measures can be taken before the spike occurs. Machine learning algorithms can be used for predicting workload. Such techniques combined with the intelligent multi-agent framework can be used to ensure that resources are automatically provisioned on demand, in an elastic manner.

1.2 Motivation

Now a days, the ability of internet and in particular social media is much increased to direct substantial traffic to web application often with a single tweet which often cases sudden spikes in the web application traffic without any notice or warning. There are many incidents that resulted in thrashing of web applications due to sudden surges in demand. Animoto experienced a jump in demand from 50 to 3500 EC2 instances in April 2008. Amazon slowed down due to massive traffic caused by pricing error on DVD boxes of US series, June 6, 2008. Sale of "The Family Guy" shot up to 200,000% exhausting the available resources in minutes. Netflix went down on Christmas Eve 2012 due to the sudden increase in load. Redix, Github and some other major websites underwent same similar situations when went down on Oct 22, 2012. Instagram and Vine got effected on August 26, 2013. There is a large list of such incidents from past where web application

experienced the similar circumstances resulting in crashing or slowing down of servers. This motivated us to develop a system that can predict sudden surges in web application traffic far enough that counter measures can be taken as despite of the elastic nature of cloud, it takes 8 to 15 minutes to make new instances ready to use for load handling which is not favorable when surge occurs in very short time. Multi-agent systems are intelligent, autonomous, and flexible and have ability to communicate with one another, all these characteristics are requirement of a clouds system. This motivated us to use multi-agent in managing cloud systems.

1.3 Problem Statement

Cloud computing is widely used for delivering and hosting services over the internet. Auto-scaling in cloud systems allows scaling up or scaling down of the resources according to the workload of the application. Many cloud service providers including Amazon, Azure etc. and softwares such as RightScale and Scalr provide Auto-scaling mechanisms. Most of the existing techniques of Auto-scaling are based on some user defined metrics and resource utilization thresholds. The current utilization of the resources is observed using monitoring tools and if the current values cross a certain threshold, then the resources like number of VMs, servers are added or released accordingly. Some recent techniques use workload trend predictions for auto scaling. These approaches mainly focus on the static allocation and do not work well with the present dynamic natured web applications where work load is highly unpredictable. This research focuses on the prediction and detection of sudden surges in load and automatically managing resources in an intelligent manners to handle that surge without compromising the performance of the application. Problem statement of this research work can be summarized as

Often events/incidents cause a sudden massive increase in web application traffic so there is a need of an automated system that can predict when a sudden spike in traffic is coming and can manage resources intelligently to handle that spike.

1.4 Methodology for Carrying the Research

Research is carried out by formulizing a hypothesis and research questions. Figure one shows the steps that we followed to complete this research work.

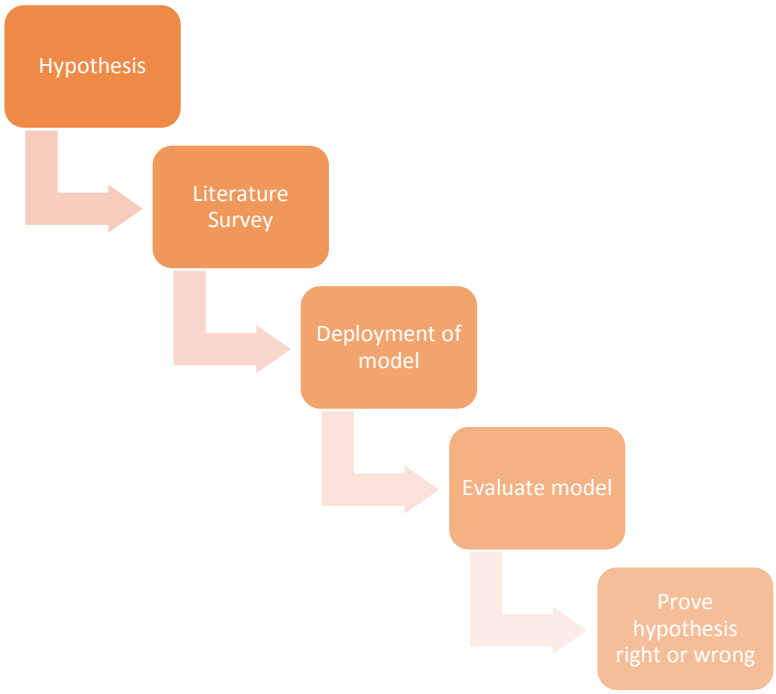

Figure 1 Steps to Carry Research

1.4.1 Hypothesis

After careful initial literature review we formulated the following hypothesis.

Sudden surges in web traffic applications can be predicted using web site's log, and managed intelligently using agents

1.4.2 Research Questions

The above hypothesis was testing by finding answers of the following research questions which we deduced from the literature.

Research Question 1: What a web log can tell us about website usage behavior?

This questions is answered by examining several machine learning techniques that use web access log, process it and predict number of requests a website may receive in future. We used linear regression, multilayer perceptron and SMO regression for experimentation.

Research Question 2: How multi-agents can help in managing resources intelligently?

To answer this question we designed and deployed a multi-agent system on Amazon cloud for our application and evaluated the performance of the application when it is managed by multi-agents instead of manual or partially automated.

Research Question 3: What strategies we can use to handle sudden surges in web application traffic?

This question is answered by using instance buffer approach to handle sudden surges which require immediate up scaling of resources in order to maintain the performance. Experiments were carried out to evaluate the performance of this approach.

1.5 Objectives of Research

The overall objectives of this research work are:

- Prediction and detection of surges in web application traffic in advance so that resources can be auto scaled before the actually surge hits
- To use multi-agents capabilities for monitoring, managing and planning to achieve better performance.
- To determine best VM instances types for various applications and choosing the best Auto-scaling plan
- To minimize the response time when web application experiences a sudden surge
- Determining the parameters that can used as an indication to describe the incoming surge

- To provide the best scaling plan when a surge is known as in case of a scheduled event.

1.6 Research Contribution

This research work will provide a multi-agent system that would free the users from specifying metrics thresholds for management of resources according to their application needs. The providers will be able to monitor and check the status of various resources such as VMs, CPU utilization to maintain performance. It provides a comparison between number of requests a web application receives and resource's utilizations metrics to determine which can better indicate an incoming surge. It also compares different machine learning techniques to predict future traffic. This will enable users to choose from multiple instance types to handle sudden traffic spikes and will help determining how the choice of instances can affect the performance of the web application at the side of end users in terms of response time and throughput. This study also reveals the impact of the history data to be used as base for forecasting the future load. The users can know how small choices like the type of OS or time window size for which predictions are made can make big impacts on performance. The last but not the least contribution of the research work is study of use of a buffer of instances that are just ready to deploy when a sudden surge takes place.

1.7 Thesis Organization

The research work is divided into five section/chapters. First chapter describes an overview of cloud systems, their usage, and motivation for this study, problem statement, objectives and contribution of this research work. Second chapters entailed an overview of the existing prediction and Auto-scaling techniques. Third chapter describes the details of the approaches and algorithms to achieve the goals of this research. Results and their relevant discussion are included in chapter number four. Conclusion and future work is discussed in final chapter of this thesis.

Chapter No. II

2. Literature Review

2.1 Auto-Scaling in Clouds-State of the art

Auto-Scaling is the ability to add and remove resources into a Cloud infrastructure based on actual usage without minimal or no human intervention needed. Currently Auto-scaling of cloud resources has become a big concern for the cloud providers as well as for clients. Dynamic resource provisioning is the key to the success of cloud infrastructure for web applications. Adding/removing resource in real time without significantly affecting the client application performance is challenging. Although the concept of auto-scaling is brilliant, current Cloud vendors are unable to perform auto-scaling operations fast enough to meet the actual requirement of the applications. For example Amazon Cloud [1] which is a one of the largest Cloud providers uses CloudWatch to initiate the auto-scaling. CloudWatch is a monitoring tool for AWS cloud resources. It provides resource utilization information such as CPU utilization, Memory usage, Network in and out, and other operational information [2]. When the monitored metrics reach a predefined set of thresholds, the auto-scaling process is triggered. Although the process seems fine but there is always a delay between when the extra capacity is need and the time when resources are actually available. This delay is caused by the launch time that EC2 instances take to start which is roughly 10 minutes [3]. Similarly Windows Azure Cloud platform has incorporated the concept of scaling using their pricing model. Users can change the amount of resources on the fly using their Management API of user portal directly. It requires customers to sit tight to observe the fluctuating demand and the capacity they can add or remove is not unlimited.

Many researchers have investigated the resource allocation problem and its solutions to meet up the cloud user requirement. The current state of the art work related to Auto-scaling of cloud resources is presented in this chapter. There are two basic approaches to manage the incoming workload from a web application hosted on cloud servers: Resource management by static load balancing and dynamic resource management.

2.1.1 Static load balancing

Static load balancing is distributing the workload between current working virtual machines to increase the performance. This is a not preemptive. The static load balancing algorithms need full information of the system in advance and load balancing decision does not depend on the current system state. The major approaches for static load balancing are round robin algorithms, threshold based and central manager based techniques. Many software and hardware solutions are available for static load balancing. Although static load balancing seems a fair solution for resource management but it does not go well when it comes to dynamic web environment where demands for resources changes very often. The process of adding new resources to system to meet new workload demands is not easy as it require manual configuration and testing which takes time. Another major drawback of static load balancing is that the workload cannot shifted to other virtual machines during execution as it is can be assigned only after the arrival of the workload.

2.1.2 Dynamic resource allocation

This type of load balancing overcomes the limitations imposed by static load balancing. The decision of load balancing is taken considering the current system state. Dynamic load balancing shifts load from over-utilized machine to under-utilized machine dynamically and addition/ removal of machines is automated. Web applications require cloud resource to be managed dynamically to satisfy the varying needs of the application and to minimize the cost. Dynamic resource allocation compliments auto scaling. The focus of our research is auto scaling. The emphasis of simple load balancing is to optimize the utilization of current resources to save energy and cost. Auto-scaling is the ability to increase or decrease the computational power in a cloud environment by either increasing number of instances or by increasing the capacity of the running instances automatically. The decision to increase (scale up) or decrease (scale down) is taken using user defined criteria or according to the behavior of web application.

According to [4] dynamic resource allocation strategies can be roughly categories into two groups based on the decision making approaches and the metrics they use. The following sections describe each group and related work.

2.2 Threshold Rule Based Techniques

In threshold rule based techniques the decision of scaling up or scaling down is taken by the cloud provider based on the threshold rules (e.g. [10], [11], [12], [13]). The future needs of the application are not considered and the decision of scaling is made using the current monitored values. Reactive solutions work in rule-condition- action manner. Rules are defined and when they met certain conditions/threshold values different actions are triggered. They are reactive resource allocation approaches of auto-scaling. Many known cloud providers use reactive auto scaling. Amazon AWS [5] uses the reactive Auto-scaling by providing Auto-scaling group SVG which consist of a number of instances. The cloud Watch monitors different parameters such as CPU, I/O and memory usage and compare it to the threshold values and the decision is taken accordingly Amazon also includes load balancing algorithms to distribute the load among instances. Many other cloud providers and brokers such as RightScale, AzureWatch, scalr etc. also use reactive solutions to achieve auto scaling [14], [15].

In [6] Chieu et al. described architecture for dynamic scaling of web applications using the number of active sessions per instance as parameter. Architecture design includes a front end Apache HTTP load balancer, a service monitor and provisioning sub-system. The scaling algorithm takes the number of active sessions as input. If the number of active sessions is above a predefined threshold the load balancer removes idle/ under-utilized instances, if the number is above the threshold then new instances are added by sub-provisioning system. Using only one feature as decision criteria can mislead and result in system performance degradation. Most popular performance metrics are the average CPU usage of the virtual machine and the response time, or the request rate. In [11] and [12] Dutreilh and Han et al. used the response time of as a threshold parameter. On the other

hand, the authors of [13] proposed their solutions using multiple performance metrics such as computation time, memory usage and network usage. They also used the correlation of these metrics as a rule.

Meiländer et al. in [7] have purposed a resource management system targeting the real time online applications such as online games and used resource usage as rule. The Real-Time Framework (RTF) consists of three main components. A cloud controller which enables communication with the cloud system, in their case Amazon EC2. A distribution controller which implements the load balancing strategy. It monitors the load values from servers and chooses between different balancing actions. ROIR server controller is responsible for implementing load balancing actions. The three actions for managing the load are user migration, resource enactment and resource substitution. Their experimental results show a noticeable decrease in the tick duration of servers. In [8] the authors have purposed an approach of a general virtualization layer to provide extra capacity. The virtualized layer sits between the physical and the service infrastructure and gives extra capacity without affecting the service workload or notifying the end users.

Threshold based solutions are very popular because of their simplicity and the fact that client can understand them easily. Although there are two main issues of the threshold based solutions. One, the choice of the accurate threshold values of metrics is very critical to the success of the techniques which is a very difficult task. Secondly they are reactive allocation techniques which means they do not incorporate the future needs of the system and result in time delay. The effectiveness of thresholds widely depend upon the workload pattern and how they tune themselves over time [9]. Threshold rule based approaches can be easily used to automate the auto-scaling process particularly in cases where applications have quite regular predictable workload pattern. However for a changing work load environment client must use the more powerful auto-scaling system.

2.3 Predictive resource allocation techniques

Predictive resource allocation suggests that the future needs of the platform is predicted and resources are allocated beforehand. Predictive techniques are widely used in weather predictions, market analysis, web site traffic forecasting, economics, and finance and in many other fields by analysts. In literature, time series and machine learning techniques are most commonly used for workload and resource usage prediction. A review of the literature reveals that predictive measures are applied for both workload and resource usage such as CPU, memory and network usage. The following sections describe multiple approaches for both workload and resource prediction.

2.3.1 Resource usage predictions

Resource usage is predicted for future using time series or machine learning techniques and capacity is added or removed from the cloud infrastructure accordingly. Simple moving average is a very simple technique used for this but it produces poor results and therefore is mostly used for providing a comparison benchmark or smoothing the time-series data [16], [17], [18]. Huang et al. proposed a prediction model based on exponential smoothing which uses both the history and current data for making predictions. They used CPU utilization and memory usage as resource metrics. They compared their model with mean and weighted moving average and results show that exponential smoothing performs better than the result of two simple methods [19]. The authors in [20] used several resource utilization features and SLA parameters as metrics to predict future usage patterns. After applying feature selection and then reduction, the important features selected as a criteria include CPU, memory and I/O utilization. Three machine learning approaches used to predict future usage patterns are Support vector machine, linear regression and neural networks. Using TPC-W benchmark datasets their results show support vector machine gives best result. The problem with these models is that they need sufficient training time to decrease prediction errors which is usually a constraint in dynamic web environment. Iqbal et al. [21] applied a hybrid model that uses reactive rules based on CPU utilization

for scaling up and for down scaling they used regression. They calculate the number of resources periodically for intervals in which the response time is within a satisfactory limit.

2.3.2 Workload predictions

Web application traffic is the feature that changes dynamically with time over days, weeks and seasons. In literature web site traffic is widely used to find the behavior, pattern and usage of application. Application workload directly indicates the change in capacity requirement that is why it is a good candidate to be used as compared to resource usage parameters. [4] Purposed a linear regression method to forecast the number of hits that a web site may receive in certain time series by using current number of hits web site's history log. They also used auto- correlation function to patterns and trends in web traffic. They have described the relationship between the numbers of web requests, cost and latency. The decision is scale up, down or NOP based on the predicted values. Their method forecasts number of hits of next few seconds which makes it ineffective as launching new VMs can take time. Their experimental results show good results for web applications where workload behavior is seasonal or follow a trend. Authors in [22] used a quadratic exponential smoothing model using real workload traces of World Cup 1998 and their results showed good results.

Another largely used technique for workload predictions is auto-regression. Kupferman et al. used auto- regression for forecasting website traffic [23]. They investigated several important parameters that effect the performance of the used techniques such as size of the history data to be used, length of the future window for forecasting. Autoregressive moving average (ARMA) is used for workload prediction in [24]. They have used Look-ahead optimization to determine the intervals when to add/remove instances. Cost is included as a key factor in provisioning decision, if the cost of adding/removing an instance is less than the SLA violation cost then decision is executed otherwise no change in number of resources. The problem with their approach is that it is not responsive to unpredicted incoming traffic as ARMA performs well with workloads follows regular patterns. The authors in [25] have proposed a block-box approach to identify the various workload

patterns and predict the capacity of web applications for different workload patterns. Based on workload predictions and Look ahead allocation model.

2.4 Surge Detection in Web application Traffic

In [26] they proposed an algorithm to predict web-server traffic in advance especially sudden spikes of traffic. Hotspot is the term used for spike or surge. Traffic is experiencing hotspot at time't' if the volume of traffic over last 'W_d' time slot satisfies:

$$\sum_{i \in [t-W_d, t]} r_i \geq H * W_d$$

Or

When the average request rate over [t-W_d, t] is at least H. r_t is the number of page requests in unit time slot. [t-1, t] is the t-th time slot. H is hotspot level i.e. maximum capacity of the server. W_d is the wide interval for which the load is computed to determine a hotspot. They used $W_d$of 3 minutes and set H to 40 for research.

Similar to [26] was the study of Baryshnikov et al [27]. They used their spline interpolation to find the load predictions of resources for SAAS architecture. They implemented their model in C#. Net and tested it in Runaware's cloud platform. Two separate modules one called trend tracker and the other load predictor. For prediction they used extrapolation predictor that takes two values tracked by trend tracker calculates the distance between them and returns the value of load at time t_1 +k. They used spline interpolation for trend analysis but not really suitable for web server traffic prediction. They used spline filter, it suppresses the average variations and stores only weighted averages of subsets of dataset.

Chandana Napagoda had demonstrated the usage of data mining techniques for website traffic prediction [28]. Chandana used SMO,linear regression, multilayer perceptron and Gaussian regression as potential techniques to predict future traffic of a website. Data set used is a web site views of 476 days obtained from Google analytics. Evaluation is gone

based on the MSE, MAE, RMSE and RMAE. The weka tool is used to generate results and results shows that SMO is the best mining approach to forecast web site traffic whereas multilayer perceptron showed worst results. In [29] they proposed dynamic window approach to predict traffic in cloud servers. They used previous data of specific time windows and the size of windows varies time to time to increase the performance of the prediction. They implemented multiple techniques of moving averages to predict future traffic and compared their results with results of predictions using statics windows sizes. They used drop-box data for experiments. Their results show that using dynamic window sizes increases performance of prediction algorithms noticeably as compared to the static window sizes.

2.5 Conclusion

The detailed study of literature reveal that many solutions exist for workload predictions in web traffic for applications where behavior follows a pattern but sudden surges are not studied widely. Most of the present solutions are based on workload predictions or user defined thresholds. Machine learning techniques are popular to forecast web site traffic for future. Literature review suggests that there are a number of user specified measures that effect the performance of underlying models. Most important of these measures is the selection of a set of accurate threshold measures for threshold based techniques. In case of prediction techniques, it is the size of the time window interval for which we need to make predictions and the size of history records to be used to make predictions. Research work is also available that examines the sudden traffic surges. Most of the work is related to detecting if the application is experiencing spike/surge during current time. Prediction of surges in future is not widely discussed. Recently multi-agents are being introduced to manage cloud environments. The conclusion to this study of literature is that there is a need to design an auto-scaling system that can cope with the varying demands of application traffic especially at time of sudden large surges in traffic.

Chapter III

3. Design and Architecture

We have implemented and intelligent multi-agent framework for predicting the surge in the workload of web applications and reconfiguring the system hosting infrastructure to meet the dynamic demands of application. The following section explain the architecture of multi-agent framework and overview of each single agent and its rule in the over framework. The algorithms used for prediction, surge detection and Auto-scaling are also briefly described.

3.1 Multi-agent System Architecture

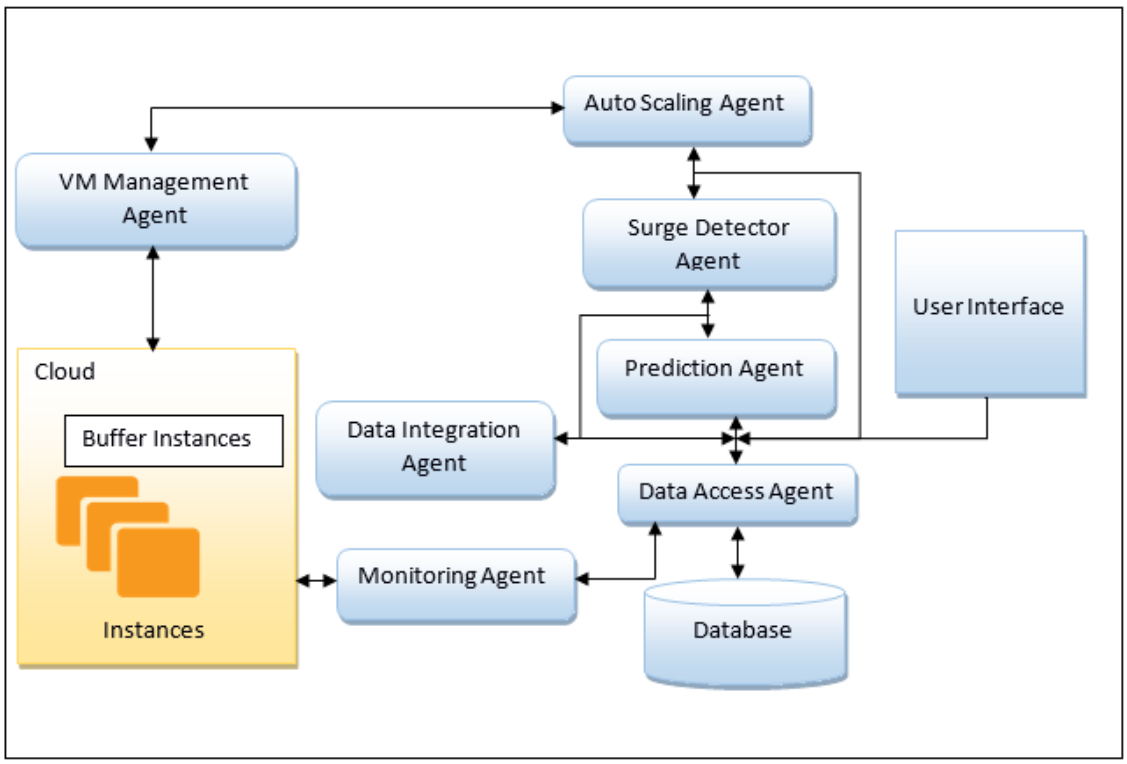

Figure 2 Architecture of Multi-Agent System

The system consists of a number of agents that have their own tasks and responsibilities and all agent communicate with each other for multi purposes. Instance monitoring agent

runs on every cloud instance and collects instances metrics like CPU utilization, memory utilization, number of requests, network in, network out etc. Monitoring agent sends all gathered data to data access agent which stores instance information into the database. Data access agent collects all data and information from all other agents and stores it to database. It also performs necessary database operations like summarization of data when required. All other agents communicate with data access agent when they need any data from database. Prediction agents plays very important role in this whole system. It collects current and past workload from the data access agent and predicts future workload using machine learning algorithms. Surge detections agent receives current and prediction workload from data access agent and determines if system will experience a surge in next time internal based on the total capacity of the running instances and future workload.

Auto-scaling agent gets the prediction workload of next interval and information from surge detection agent and plans out how many and which instances to launch to accommodate the upcoming surge. It is responsible for making important decisions such as up scaling, down scaling and chooses which instances to launch or shut down from available pool of instances. Auto-scaling agent is also responsible for managing the instance buffer. It updates buffer periodically and also when instances are launched from the buffer. Data integration agent collects current workload and future workload from the data access agent and compares both of them which helps analyzing performance of prediction agent. VM management agent receives the scaling plan from Auto-scaling agent and launches or shut downs instances. A user panel is also introduced which is used by clients to specify any incoming event which may result in sudden spike in application traffic. Clients can give numbers of instances to be launched on a specific date and time or they can give an approximated number of requests that application may receive. The Auto-scaling agent makes Auto-scaling plan to handle the expected workload and VM management executes that plan. Next sections of this chapter describe working of each agent in detail.

3.2 Monitoring Agent

The job of the monitoring agent is to gathers the information about the current system state. A separate monitoring agent runs for every instance that is currently running. It collects instance data via Amazon Cloud Watch which is a tool for monitoring the state of instances. It collects CPU utilization, network in and out and access logs for every minute from Cloud Watch and sends all this information to data access agent. This data is then further used by prediction, surge detection and Auto-scaling agent.

3.3 Data Access Agent

Data access agent manages the database. It receives all the data from other agents and stores it to MYSQL database after processing. Any agent that wants some data for its processing sends a request to data access agent through jade API and it provides data after fetching the database. It maintains separate database tables for multiple types of information such as access logs, CPU utilization, future predictions, scaling plan details etc. It maintains and updates the database tables when new data is received. It sends request logs of previous time interval to prediction agent which after training forecasts number of requests for next time interval and sends this information back to data access agent. Data integration agent communicates with data access agent to get current request rate and predicted request rate per minute and compares both to represent current picture of the system state.

Data access agent maintains profile for every instance type which stores the maximum capacity of instance in terms of maximum CPU utilization and maximum number of request it can process without exhausting. This information is used by Auto-scaling agent to make appropriate scaling plans.

3.4 Prediction Agent

Prediction agent predicts workload for future using the workload of past. It gets the number of request received per minute for recent past from the data access agent use that data to train machine learning techniques and then predicts number of requests for future.

Statistical models such as autoregressive moving averages, autoregressive models, and moving averages and are used for forecasting the time series [30]. They use history data and work by pre-processing that data and do well in finding patterns over time. Due to the dynamicity of the web applications pre-processing website traffic logs can sometimes hide important information for example a surge may be removed as an outlier during the pre-processing which is a useful piece information. There is a need of some prediction technique that will make use of the unusual information instead of omitting it. Prediction agent uses machine learning techniques for workload forecasting because their accuracy is high in the presence of outliers or missing values. Another reason for using machine learning algorithms is that they predict future by learning from training data and adapt to variability of data over time. We have used three learning algorithms Multilayer Perceptron, Linear Regression and Support Vector Machine as they are particularly suitable for numeric prediction involving times series analysis.

1) Multilayer perceptron: This is a neural network based algorithm in which large numbers of units are connected to each other in a similar manner to the human brain. It can be trained without specifying any particular function up front [31].

2) Linear regression: Is a statistical method which develops relationship between dependent and independent variables. Linear Regression works by building a prediction model/regression equation to fit the given data. This model then predicts target values for unknown data [32].

3) A Support Vector machine is a form of supervised learning. It represents the training data as data points in space and each data point falls into a category of examples separated by gap. New data points are mapped into the same space and their category is determined by which side of gap they fall [33].

Prediction agent uses data from recent past to train the machine learning algorithms. Training is done only using recent data instead of large data sets because over large time span variation in workload patterns get suppressed. We have only used data of last time

window. The size of the predicted time windows is also kept small to 10 minutes. Prediction agent adjusts the size of the training data to improve the accuracy of the algorithms. Predictions are made every 10 minutes to make the system more robust towards sudden surges. The accuracy of three techniques is compared and prediction agent continuous with the one whose accuracy is better among the three over time. It sends the predicted workload for the next time window to the data access agent. Data access agent stores the results in relevant table and sends them to Auto-scaling agent to make a plan to adjust the predicted load.

3.5 Surge Detection Agent

Surge detection agent analysis the current and predicted traffic received by the application and determines if this can be considered as a surge which is also named as Hotspot in literature by researchers. Surge or hotspot is the sudden increase of traffic over a short time. Traffic is experiencing hotspot at time‘t’ if the request rate r_t over last ‘W_d’ time intervals satisfies:

$$\sum_{i\in[t-W_d,t]} r_i \geq H * W_d \qquad (1)$$

Where

r_t is the predicted number of requests in a specific time period.

H is hotspot level i.e. maximum of the platform capacity in terms of number of requests that it can handle and without violating SLAs. It is determined by aggregating the individual capacities of the current running instances and capacity of each instance type is calculated by profiler agent.

W_d is the wide of interval for which the load is computed to determine a hotspot. We set future time interval to ten minutes and the time window size is set to 2. Surge detection agent determines whether the application will undergo a surge using algorithm 1.

Algorithm 1 Surge Detection Algorithm

Input:

Predicted request rate for time interval t
Total capacity H of the platform

Output:

Status of surge- surge detected/ no surge
Variation of Request rate from total capacity

Start

Var Surge
Var Variation

If (RequestRate > Total capacity H of the platform)

Surge= True

Variation = RequetsRate – Total Capacity H

Return Variation

Else

Surge= False

Return Surge

End if

End

It starts with comparing the predicted request rate for time t with the total capacity of the platform. If the request rate is greater than the platform can bear then surge in detected and surge detection agent generates an alarm and the difference between the capacity and request rate is determined. The surge detection then sends this difference to Auto-scaling agent which computes the number of instances that are need to accommodate difference. If no hotspot is detected then a message is also generated showing there is no incoming surge. Surge detection agent also observes the nature of the surges as they differ in size and nature from application to application. As an example a surge on a social media web site caused due to some viral video usually lasts longer and a surge in traffic of a news channel covering some event live will last for a few hours. Two important factors that are; how quickly a surge comes and goes away and how much the traffic rate is increased compared to the average normal traffic rate, are observed. Taking into account these two

factors that we deduced from observing the workload patterns of the past surges we have categorized surges into three types. The reason for choosing these factors is that these two are the powerful features that differentiate between surges of different types of web application.

3.5.1 Small Surge

A small surge is the surge of web application traffic that lasts for a short duration and goes away as quickly as it comes. For instance a surge caused by the tweet of a celebrity or a post on Facebook. It caused a comparatively small but sudden increase in traffic and the traffic goes down quickly.

3.5.2 Large Surge

A large surge is the sudden massive increase in traffic of a web application. A large surge can be both short lived and may be long lived. For example live coverage of some hot event may cause a sudden massive increase in the traffic. However, traffic will come to normal soon after the event ends. On the other hand surge caused due to the release of a new product or a sudden sale will last longer. A large surge causes the traffic to increase many times than the normal traffic an application receives.

3.5.3 Medium Surge

A medium surge is characterized by an increase in traffic of an application which is gradual. The traffic an application receives increases gradually from normal rate to a much higher rate and then gradually comes back to normal. For example traffic increase on a sports website due to some seasonal championship. Medium surges usually lasts longer than small and large surges.

3.6 Auto-scaling Agent

Auto-scaling agent is the back bone of the entire system as it determines how to handle the incoming surge in workload. It receives information about the current capacity of the system, current request rate and predicted request rate from data access agent and uses that

information to determine the number of instances that are required to meet the predicted demand. It makes decision of which instances to launch and shut down when needed. It is also responsible for the maintenance of the buffer. Auto-scaling agent updates buffer periodically and whenever instances are launched from the buffer. Instance buffer is maintained to keep the system ready to handle sudden spikes of workload. Buffer contains multiple instance in ready state to save the time needed to boot a new instance. Size of the buffer is varied as the request rate varies over time.

Figure 3 explains the working of the auto-scaling agent. It collects the predicted workload for the next time interval and current capacity of the system from the data access agent. It computes the required capacity for handling the workload and the difference between current and required capacity. If the required capacity is greater than the current capacity, auto-scaling agent computers the scaling plan to adjust the extra workload by choosing instances from instance buffer and sends its plan to database. Data access agent sends new computed plan and sends message to VM management agent to launch new instances. Amazon provides various types of instances with various CPU, RAM, and storage capacities according to their cost. Clients can choose the instances according to their application needs. Computing number of instances is easy when all the instances are of same type and capacity and have same cost but it becomes a challenging task to choose best instances for the current and future workload. Auto-scaling agent takes into account various factors while choosing new instances. Similarly down scaling plan is also computed considering various factors like remaining time etc.

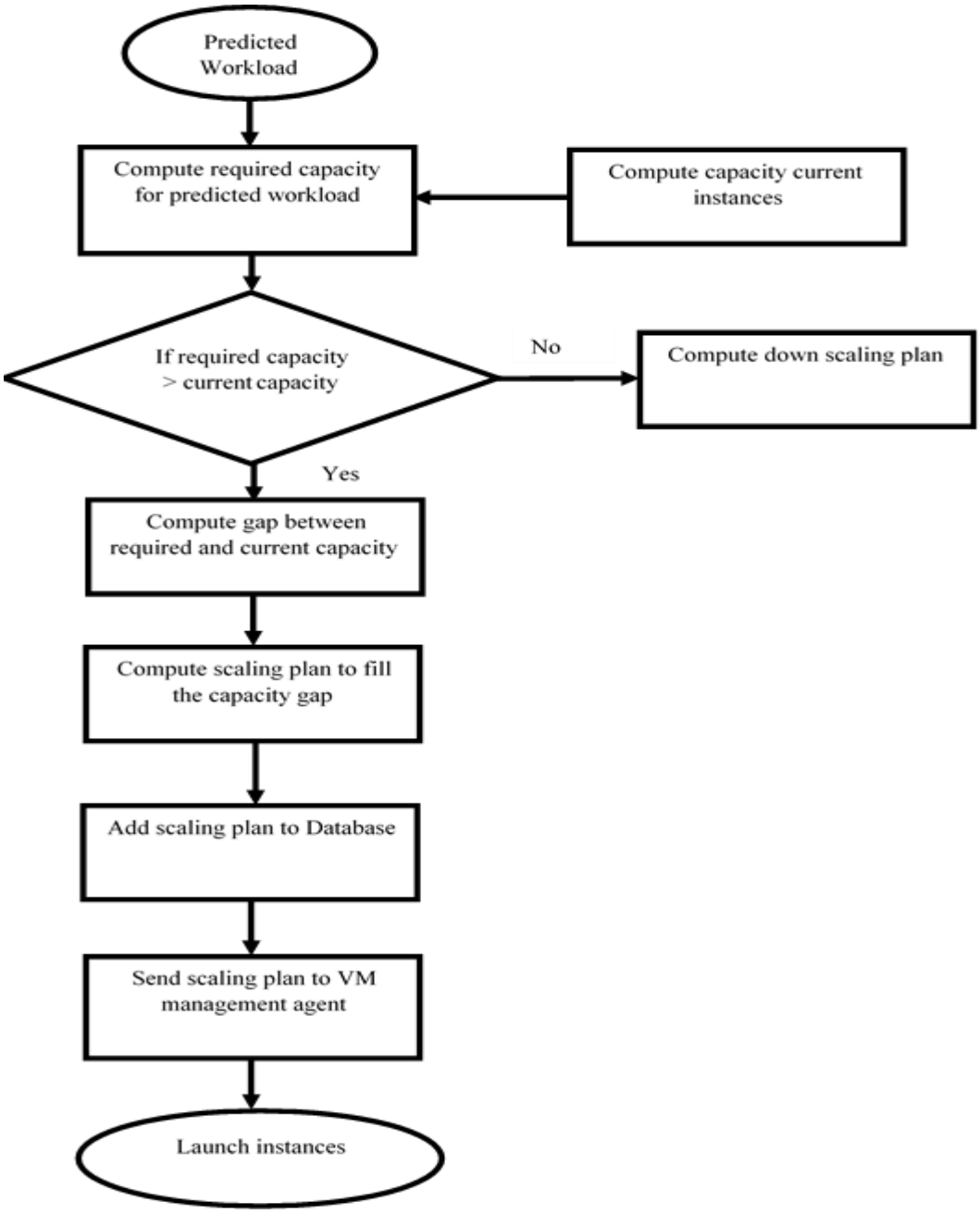

Figure 3 Flow diagram of Auto-scaling agent

3.6.1 Up Scaling of Resources

Auto-scaling agent scales up the instances when current capacity is less than the required capacity to handle the predicted surge. It computes the difference between the current

capacity and required capacity and then finds the best instances to launch. Auto-scaling agent considers following factor while choosing new instances.

1. CPU credit of the instance
2. Prefers smaller instances over large ones
3. Choose those instances that minimize the difference between demand and capacity

For the research purpose we have used only three types of Amazon instances that are T2.mirco, T2.small and T2.medium. Specifications of these instances are given in table one.

Table 1 T2 instance types and their specifications

Instance Type	**vCPU**	**Memory in GB**	**Price/hour**
T2.micro	1	1	0.013$
T2.small	1	2	0.026$
T2.medium	2	4	0.052$

T2 instances provide a baseline level of CPU performance with the ability to burst above that baseline level. The baseline performance and ability to burst are governed by CPU credits. CPU credits can accumulate for up to 24 hours when instance is idle. When an instance with more CPU credit is used then the CPU performance increase. That is the reason auto-scaling agent prefers the instances with large CPU credits. Baseline performance and CPU credits earned per hour are given in table two. Consider two micro instances one with 30 CPU credits and the other with 60 CPU credits. The performance of the second one would be double as compared to the first one. Auto-scaling agent also prefers smaller instances over larger ones as small instances cost less and take less time to reconfigure.

Table 2 CPU Credits of T2 instances

Instance type	**Initial CPU credit***	**CPU credits earned per hour**	**Base performance (CPU utilization)**	**Maximum earned CPU credit balance**
T2.micro	30	6	10%	144
T2.small	30	12	20%	288
T2.medium	60	24	40%	576

3.6.2 Down Scaling of Resources

When required capacity is less than the current capacity it means some of the instances are underutilized. Auto-scaling agent looks for the underutilized instances and shut downs the one which has less remaining time.

Remaining Time = Current time of instance – Start time of instance

Down scaling of instances is explained in figure 4. Auto-scaling agent does not immediately terminates the instance. It schedules the termination right before the start of the next hour. It sends the information of the underutilized instances to the data access agent and then instance is used for maintenance activities.

3.6.3 Instance Buffer

Buffer is maintained to keep up with the dynamic changes in demand. All instances in buffer are always in ready state. When a sudden burst is observed in workload they are immediately added to the load balancer to distribute the load. To keep the cost low we used maximum four instance as buffer. Buffer size is an important choice to make. Client can specify a fixed sized or variable sized buffer according to their application demands. We used a variable sized buffer which is updated every 20 minutes.

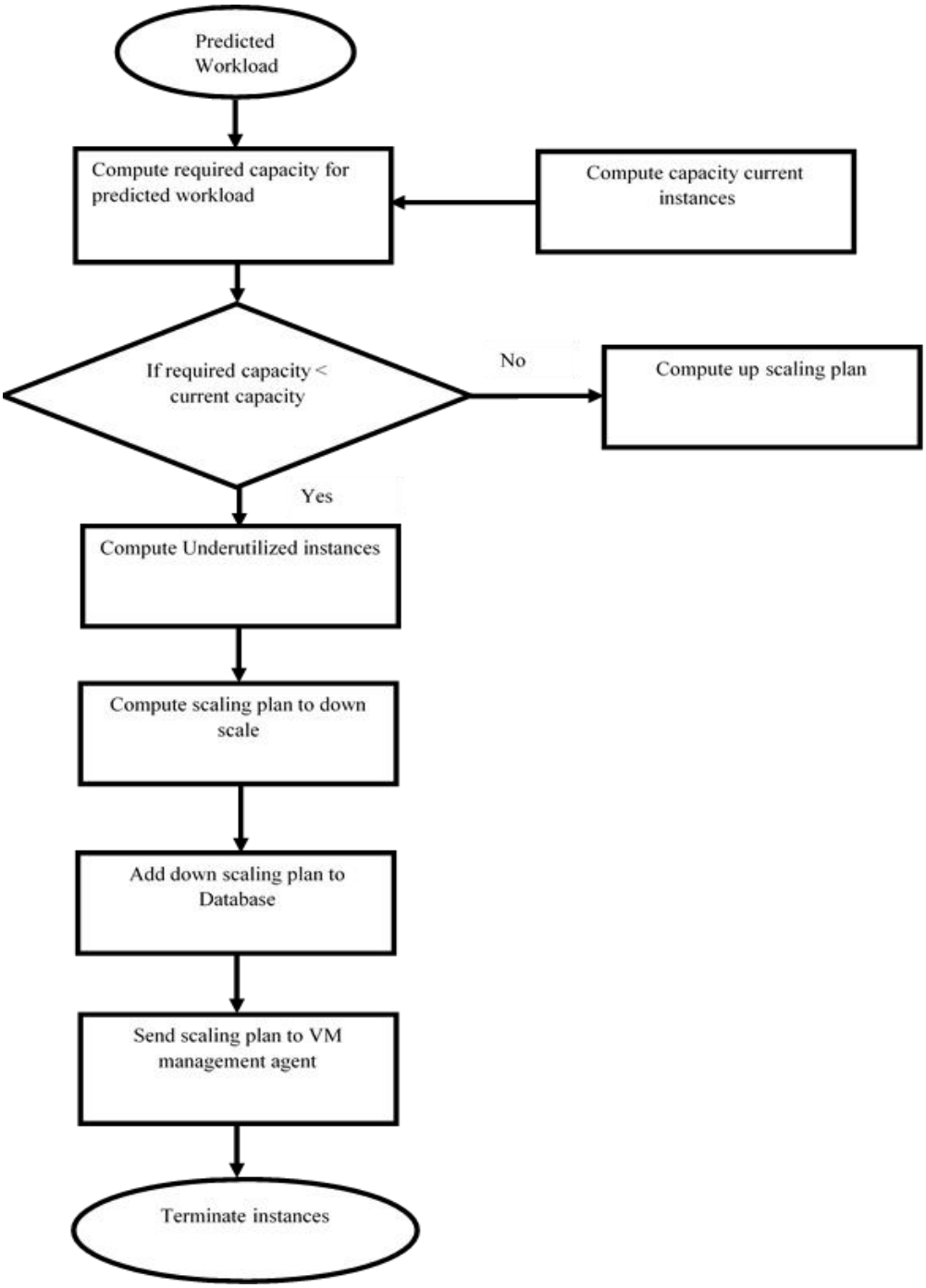

Figure 4 Flow Diagram for Down Scaling of Instances

We have used instance buffer of variable size and auto-scaling updates buffer regularly. Algorithm two illustrates the process of updating the buffer. Algorithms computes the current number of instances and capacity of the buffer. If the size of buffer is less than

Minimum number of instances that buffer must contain at least then buffer is updated otherwise buffer is not updated and auto-scaling agent sends current buffer size to data access agent which updates the BufferInsatnce table in database. Maximum and minimum number of instances a buffer can have varies from application to application. A large organization may need to maintain a large number of instances to provide the best performance to their customers even at times of spikes according to the previous sudden spikes their web application faced while buffer size of an application owned by an individual limited by the surge size and cost of the instances.

Algorithm 2 Buffer Update Algorithm

Input:

Buffer size

Total capacity of the buffer

Average Request Rate for last two time windows

Output:

List of instances present in buffer

Start

Var MaxBufferSize = 4

Var MinBufferSize = 2

If (MinBufferSize < 2)

Compute capacity required for average request rate of last two windows

If (Required capacity > Total Capacity)

Compute number of instances required

Add instances to buffer while MaxBufferSize =4

Return Instance List

Else

UpdatedBufferSize = Buffer Size

Return Instance List

End if

End

3.7 Data Integration Agent

Data integration agent collects current workload metrics monitored by monitoring agent and the workload predicted by the prediction agent and presents a real-time comparison of the two in the form of graphs. This real time picture helps to monitor the accuracy of the other agents. It uses JChart API to present real time graphs of request rate, CPU utilization, memory utilization and network in and out. It also checks the accuracy of the surge detection agent by examining the number true and false alarms generated. Data Integration agent communicates with Data access agent through jade API and all the data is encoded in to JSON format. It then inserts all the computed correlations values in the 'monitoringdatapattern' table of MYSQL database with their timestamps which is used for seeing the relationships between different parameters and for the detection of hotspot. For accurate prediction of hotspot there are many other techniques which are implemented

3.8 VM Management Agent

VM Management agent is the agent responsible for executing the auto-scaling plan. It receives the auto-scaling plan from data access agent and reconfigures the platform by adding instances to it or by terminating the instances according to the demand of the application traffic. It uses Amazon API to launch or terminate the instances. VM management agent is also responsible for adding or removing instances from the buffer. It uses the information from the data access agent in a cyclic manner. It also sends the data such as ID, VM type etc. of the instances it launches are shuts down to the data access agent to be stored in the database.

3.9 User Interface

Not all the surges an application receive are unpredictable. Sometimes clients know that an upcoming event can lead to a potential surge for example the launch of a new product, sales or discount on e-commerce website. A user can plan out such events using user interface. There are many options available for the user to indicate the potential surge. The user can specify the start and end date and time during which application may experience

a spike. User can indicate the approximate duration and size of the surge. He/She can also schedule the launch of the new instances at some particular date and time. All these options enable user to tell the system about a surge even when the user has minimum information about it. User inputs are gathers by the data access agent and are stored in the database. Data access agent sends the used provided information to auto-scaling agent and its plans out the scaling accordingly.

Chapter IV

4. Results and Discussion

This chapter represents the evaluation of results of the proposed multi-agent system. All the experiments were carried out carefully using real time data of Wikipedia. The agent platform was deployed on Amazon. The results of prediction, surge detection and auto-scaling agent are presented and discussed in detail in different sections of this chapter.

4.1 Data and Environment Setup

The evaluation was done by deploying the media wiki application on the Amazon Cloud. Wikipedia access traces of January 2008 were used for experimental purpose. These traces are used by many researchers as they are complete and their access logs are also available. Wikibench tool was used to deploy Wikipedia application. One of the many reasons for using Wikibech is that it simulates real traffic over real application instead of using synthesized workload over some "toy" application. Wiki tracer was used to simulate the workload from January 2008 for Wikipedia. Figure 5 shows the workload generated that is used throughout the research for different experiments. WEKA API for java was used to implement machine learning algorithms. Prediction algorithms were applied to the data using the different options available in Weka. The workload contains three different surge types at different times. Underlying database was SQL and Jade is used as agent platform. The application was deployed on Amazon for a short period to monitor the performance of the system. Only three types of Amazon instance i.e. t2.micro, t2.small and t2.medium were used for experimental purpose. The VM management agent used a golden image to launch new instances using Amazon API.

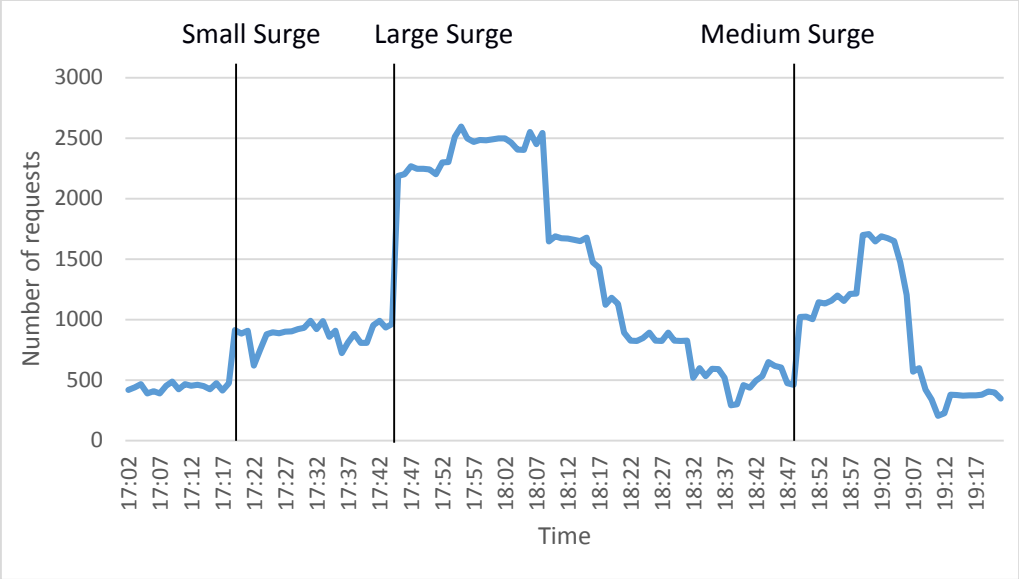

Figure 5 Wikipedia Workload

4.2 Workload Predictions

Linear Regression, Multilayer Perceptron and Support Vector Machine were used to predict future workloads using the past access logs. All of these are machine learning techniques and they use past data to learn about the future trends. Their continuous learning ability makes them a good candidate for the dynamic environment of web traffic where data show large variability over time to time. The workload generated contained dynamic changes to evaluate performance of auto-scaling algorithm for different types of surges. We only need to make predictions for the near future therefore the size of the sample used was quite small because a surge often happens hastily and only lasts for few hours. In this case data of only last few hours will show an increase in traffic and can perhaps lead to an incoming surge. Use of large data sets which include data from days to weeks or months can misguide results. Long term usage patterns of website traffic can be identified by analyzing large volumes of past data. A very important factor in prediction is the time window size for which we want to make predictions. In our case we used two different window sizes to show their impact on results.

4.2.1 Workload Predictions for Small Sized Surge

The workload was generated such that it can show multiple types of surges. A small surge is characterized by a small sudden increase in application traffic. In case of small surge usually total capacity of the cloud infrastructure is relatively small limited to two to three number of instances. The average request rate in this case in approximately 550 requests per minute. Figure 6 shows the prediction results for multiple algorithms when traffic experiences a small surge. Machine learning algorithms builds a model by training on past data, and then predict future workload using trained model. Support Vector Machine and Multilayer Perceptron perform almost in similar manner with a very small difference in their predicted number of requests. Linear regression performs slightly better than the rest two techniques. Initially the workload was kept smaller to demonstrate performance of predictions algorithms when average request rate varies from 200 to 1000 requests per minute. In this scenario the request rate during surge was also small and thus can easily be handled by adding 1 to 2 instancing.

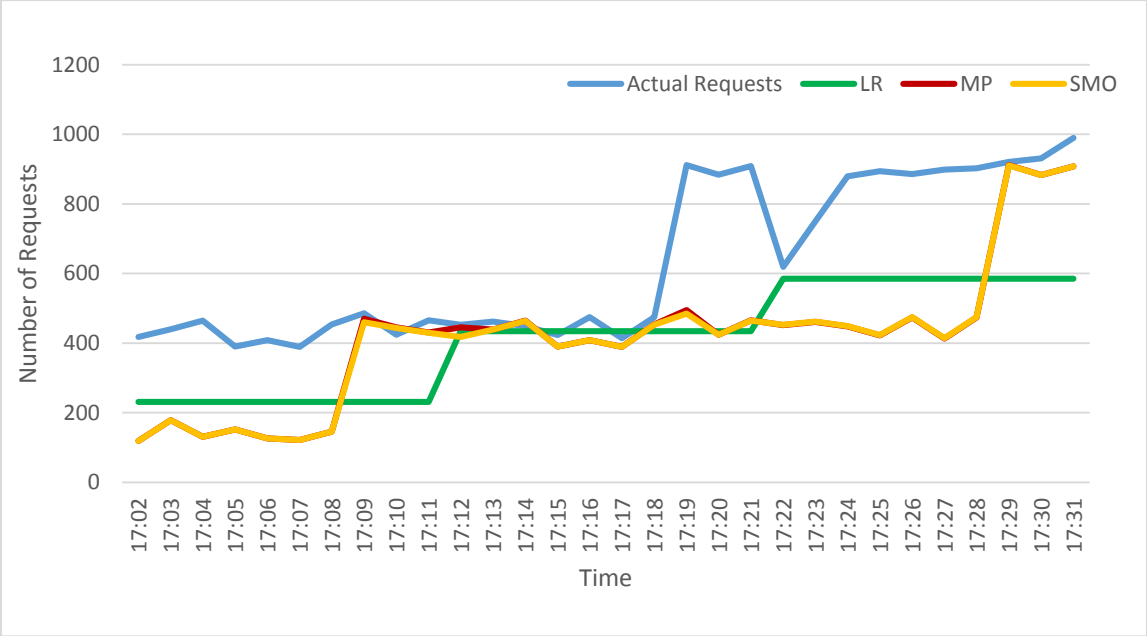

Figure 6 Comparison of Actual and Predicted number of requests by Machine Learning Algorithms for Small Surge

Algorithm's accuracy was measured in terms of Mean Absolute Error, Root Mean Squared Error and Relative Absolute Error. Table 3 describes the accuracy of each technique.

Mean Absolute Error (MAE): It shows how close the predicted values are to the actual values. It measures the average absolute difference between predicted and actual values.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} | f_i - y_i | \quad (2)$$

where f_i is the prediction and y_i the actual value. Root Mean Squared Error (RMSE) computes the accuracy in the form of sample standard deviation of the differences between predicted values and actual values.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (ft - y)^2}{n}} \quad (3)$$

Relative Absolute Error (RAE) measures how large the absolute error is compared with the predicted value. Linear regression gives better results as compared to the other two techniques as can be seen in table 3. Predictor agent compares accuracy measures of all three algorithms and chooses the one with minimum error in last time windows. The values of these measures are the number of requests per minute.

Table 3 Summary of Accuracy Measures of prediction algorithms for small surge

	MAE	RMSE	RAE
SMOreg	221	281	453
Linear Regression	220	259	449
Multilayer Perceptron	221	280	454

Mean Absolute Error of 221 means the average difference between actual number of requests per minute and predicted number of requests per minute in 221 i.e. actual request rate is 880 requests per minute and predicted request rate is 660. All three algorithms show average performance in this case. The reason for this is that machine learning algorithms

train and learn over time so initially we observe an average but still acceptable performance.

4.2.2 Workload Predictions for Large Sized Surge

Figure 7 illustrates the predicted number of requests for a large surge. The workload was increased from an average of 900 requests to 2000 requests per minute. During the large surge the number of resources needed also increase suddenly. The graph shows that machine learning techniques adapt quickly to the sudden changes in pattern. Surge was observed at 17: 45 when number of requests shoot quickly and machine learning algorithms adapt themselves quickly to this sudden change and predicted number of requests also increased. The predicted request rate shows the change at 17:52 minutes.

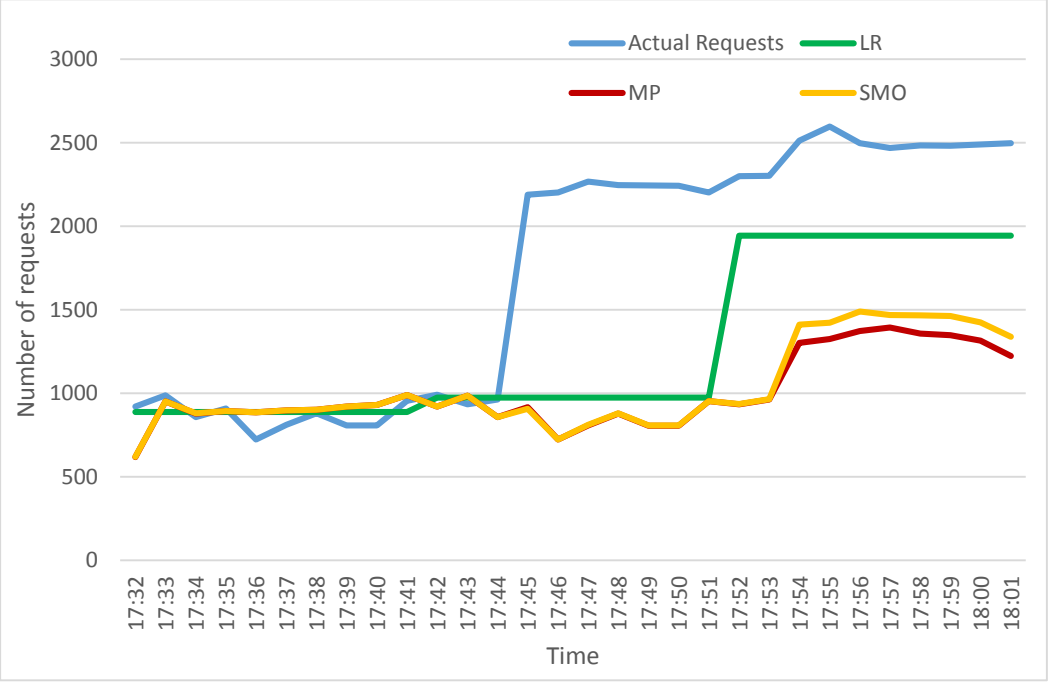

Figure 7 Comparison of Actual and Predicted number of requests by Machine Learning Algorithms for Large Surge

Numeric summaries of results when a sudden large surge is observed is given in table 4. Accuracy of linear regression is better than the SMOreg and multilayer perceptron therefor prediction agent chooses the predictions made by linear regression to send to auto-scaling agent. It can be seen that the accuracy of the algorithms was improved over time as they learn and train themselves over time from past data.

Table 4 Summary of Accuracy Measures of prediction algorithms for large surge

	MAE	RMSE	RAE
SMOreg	765	833	729
Linear Regression	454	519	100
Multilayer Perceptron	765	831	728

4.2.2.1 Workload Predictions during a Large Surge

Figure 8 represents the prediction results when application is experiencing a surge in traffic. The lines for actual and predicted request rate are close to each other showing the overall improvement in predictions. From figure 7 and 8 it can be seen that machine learning algorithms learn quickly. When surge occurred initially the algorithm's performance is average but it learns quickly from the most recent training data and its accuracy improves from average to very good. Figure 8 shows the improved performance of prediction algorithms within a time span of minutes.

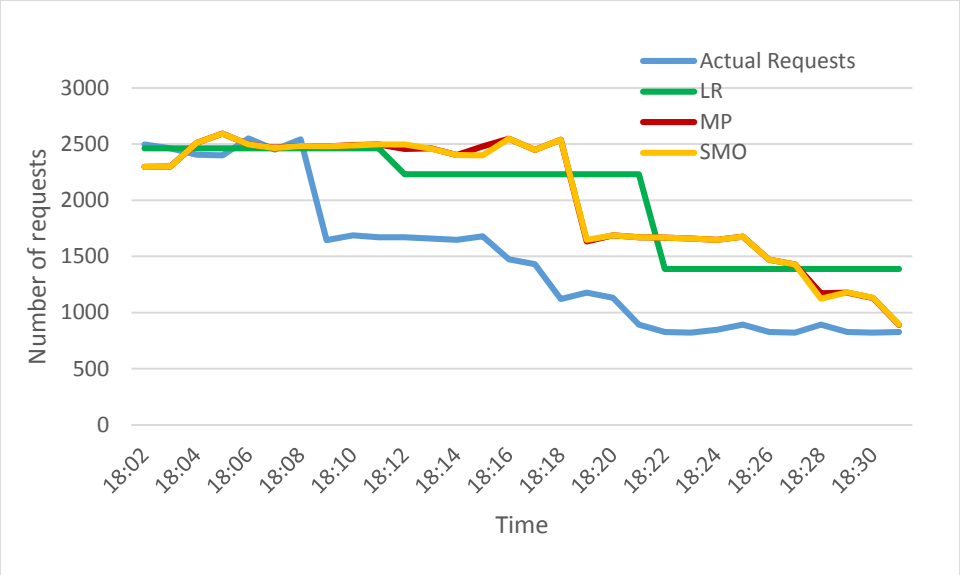

Figure 8 Comparison of Actual and Predicted number of request during a surge

Accuracy of linear regression, SMOreg and multilayer perceptron is given in table 5. The accuracy of the prediction techniques was improved as compared to the accuracy when application was undergoing a large change that resulted in surge. The effect that is visible form graphs can also be seen in table 5 numerically. The value of Mean Absolute error was decreased from 454 to 304 for Linear Regression and similar change can be observed for SMO regression and multilayer perceptron.

Table 5 Summary of Accuracy Measures of prediction algorithms during surge

	MAE	RMSE	RAE
SMOreg	345	378	87
Linear Regression	304	319	79
Multilayer Perceptron	348	371	87

4.2.3 Workload Predictions for a Medium Surge

After undergoing a large swing in workload, we decreased the workload to monitor the performance of prediction agent. The workload is later increased to observe a medium sized surge. The workload was initially decreased to a lower value of approximately 400 average requests per minute and then the workload was gradually increased from 400 to 1600 requests per minute. The results are represented in figure 9. The graph shows that linear regression is more robust towards changes in workload and adapts more quickly as compared to the SMOreg and multilayer perceptron. Numeric summary of predictions is given in table 6.

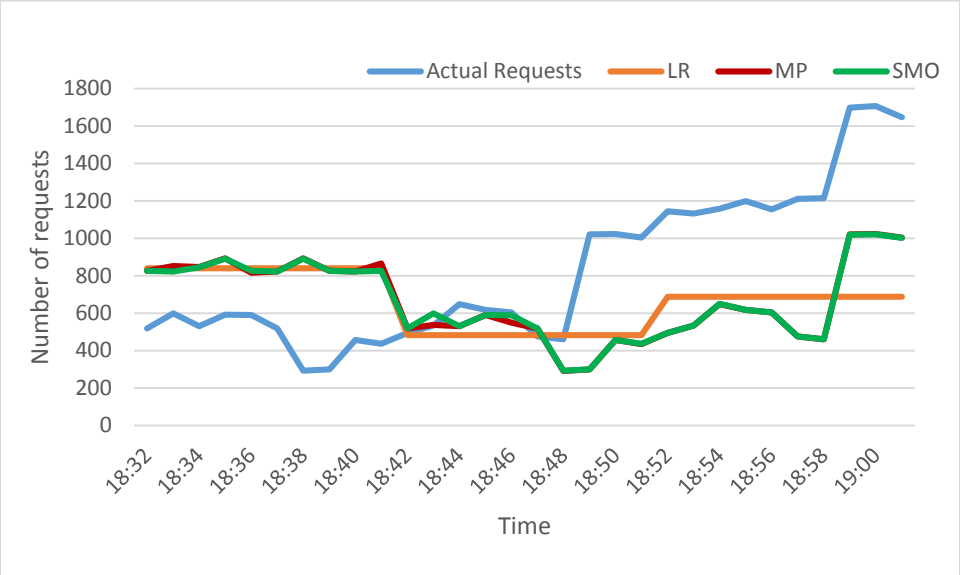

Figure 9 Comparison of actual and predicted number of requests medium surge

Table 6 Summary of Accuracy Measures of prediction algorithms for medium surge

	MAE	RMSE	RAE
SMOreg	302	352	156
Linear Regression	278	290	129
Multilayer Perceptron	306	367	160

Figure 10 shows the summed actual and predicted no. of hits over the time period equal to the size of time window i.e. for ten minute time slot. The difference between actual and

predicted number of requests is relative to the average number of requests of the application. A web application with a small average request rate will indicate a very small error in terms of number of requests when accuracy of prediction technique is 95%. While the error value for an application where average request rate is very large, the same prediction technique will generate large error value because the 5% of 100 is only five but 5% of 1000 is 50. That is why difference between actual number of requests and predicted requests increases as the total requests received by the application increase. However this difference can be overlooked for the same reason. Putting this into other words, when request rate is large i.e. thousands of requests within a specified time window, a difference of 500 is not that important but same value matters when requests is in hundreds.

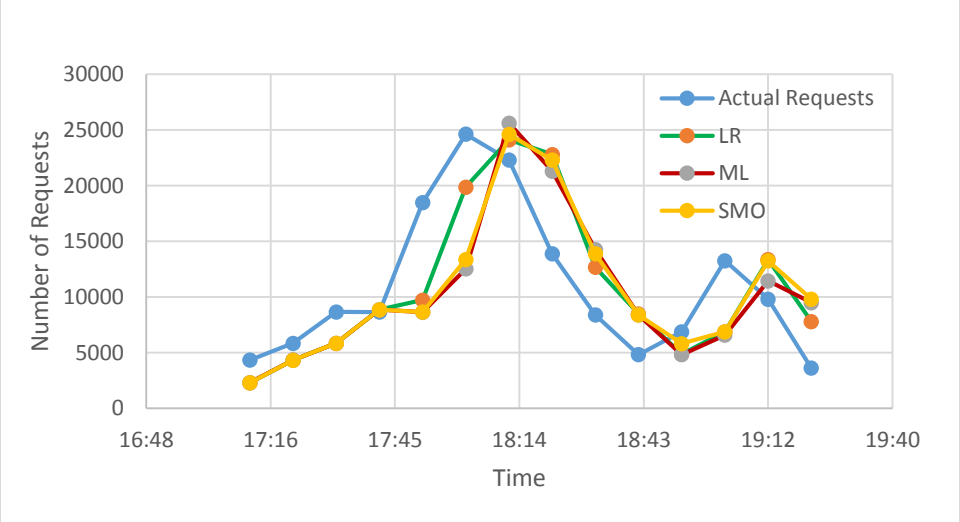

Figure 10: Total number of Requests during Time Window of Ten Minutes

Ten minutes time window size was chosen very carefully after performing many offline prediction tests. The offline prediction results are included in the appendix. Offline experiments reveal that the performance of the prediction algorithms is greatly affected by choice of time windows for past data-how much past data to use for future predictions and also the size of future time window for which the predictions are made. Choosing data from

recent past produces better results as compared to when all available data is used for training the machine learning algorithms. In the same manner smaller the size of the future time window the better the results are. But choosing too small past data set or making prediction for very short time periods compromises the performance of prediction algorithms.

4.3 Surge Detection

All instances being used in experiment were first profiled using profilers' agent. The profiler got benchmarks/ thresholds for multiple parameters including maximum number of requests an instance can handle without exhausting, maximum CPU utilization, maximum memory Utilization etc. These parameters were gathered by making some assumption; lowing the thresholds to a certain limits as for experimental purpose it is a really difficult task to generate such large traffic to reach these points. All the Thresholds are described in table 7.

Table 7 Threshold values for Various Instance Parameters

	T2.micro	T2.small	T2.medium
Max Capacity H	247	306	464
Max CPU utilization	43%	37%	34%
Max memory Utilization	54%	40%	30%

The predictor agent makes predictions after every ten minutes interval for next ten minutes. The surge detector agent gets predictions results and checks if there is an incoming surge in next time window of ten minutes. It generates a tick every ten minutes Surge detector then analysis the predicted number of requests for next time window to see if it can lead to a surge and if a surge is detected if generates an alarm which then triggers scaling agent to calculate the number of instances to handle the predicted surge and the instances are launched by the planner agent before the surge actually arrives. Table 8 shows the totals

number of ticks generated and their status whether they were true of false alarms. In table "Alarm" means the number of times a true alarm was generated i.e. detection agent generated an alarm that a surge in detected and there was a surge in actual. "~Alarm" indicates false alarms. When an alarm was generated but there was no surge. Similarly "Surge" indicates that a surge was detected and there was an actual surge in traffic. "~Surge" indicates that a surge detection agent detected a surge but there was no actual surge in the traffic.

Surge detector generated a total of 14 ticks periodically in search of surges in future traffic. As a result seven alarms were generated by the surge detector agent and all were true alarms. Out of 14 ticks seven ticks didn't detected any surge therefore no alarms were generated, however out of these 7 ticks there was one surge that surge detector agent could not identify. It didn't generated any alarm although there was a surge in traffic.

Table 8 Alarms generated by Surge Detector

	Surge	~Surge
Alarm	7	0
~Alarm	1	6

4.4 Auto-Scaling

Auto-scaling agent gathers the data from prediction and surge detection agents and makes the scaling decision using that information. Figure 11 shows the overall request rate and total capacity of the currently running instances. It can be seen from the graph that smaller surges appear at the start so capacity keeps adjusting to handle these surges. A big surge takes place at about 17:45 minutes and lasts for almost 24 minutes. Another quick and short lived spike occurs around 19:00. Scaling agent looks for appropriate scaling plan when it receives alarm generated by surge detector. It calculates the number of instances needed to

accommodate incoming surge from the available instances from the buffer. It also updates buffer every twenty minutes to maintain a reasonable sized buffer to use when needed. It prefers the instances that decrease the difference between the sum of predicted workload and the total capacity of the system. If more than one combination of different instance types is available within the buffer it favors the ones with max CPU credit. CPU credit becomes an important factor to increase the performance of the application when we use buffered instances. In buffer instances are in ready state but they do not have any actual CPU utilization as they are not added to load balancers so their CPU credit accumulates over time. The instances with greater CPU credit are added to the load balancers and increase performance effectively.

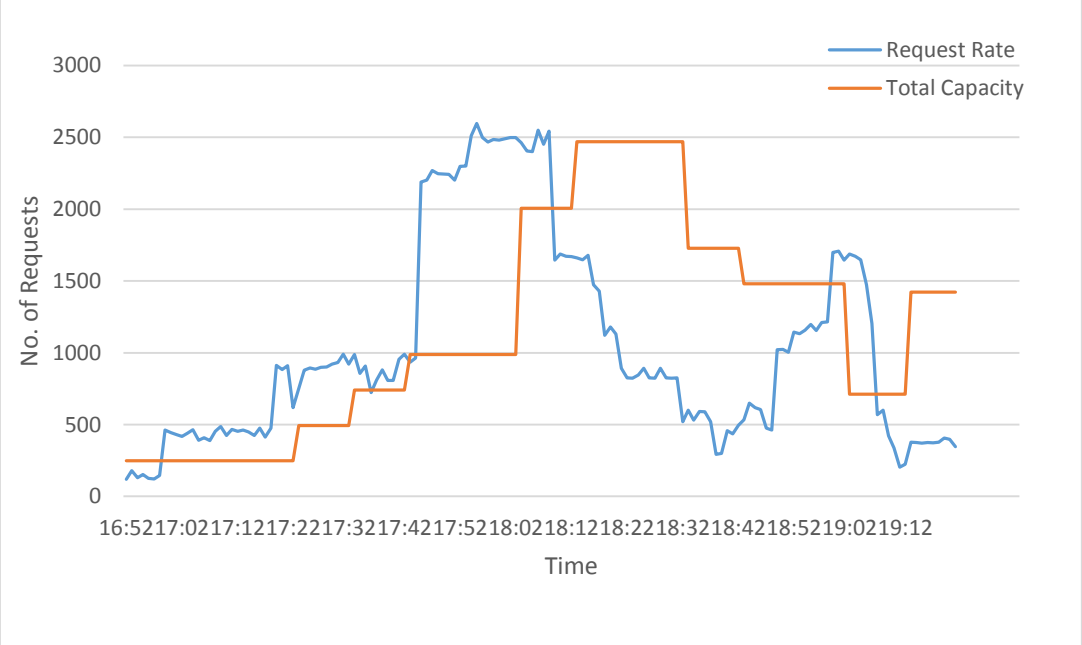

Figure 11 Overview of total number of requests and total capacity of running instance at the moment

Figure 12 shows the adaptability of the instances according to the incoming load. The lines for capacity and demand overlap at number of times showing that actual capacity is very close to the actual demand. A very visible factor between our auto-scaling technique and conventional techniques is the decreased time gap between availability of resources and

the actual workload demand. In conventional auto-scaling techniques the application suffers from under performance and over utilization of resources for almost 23 minutes from 17: 45 to 18: 02 before new VMs are available to meet the demand. While in our auto-scaling technique this time gap is decreased to 7 minutes because of the usage of buffer resources. At few times instances are underutilized as seen by graph. These are the instances that were launched by the planner agent at peak times now when the load is decreased they are scheduled to shut down.

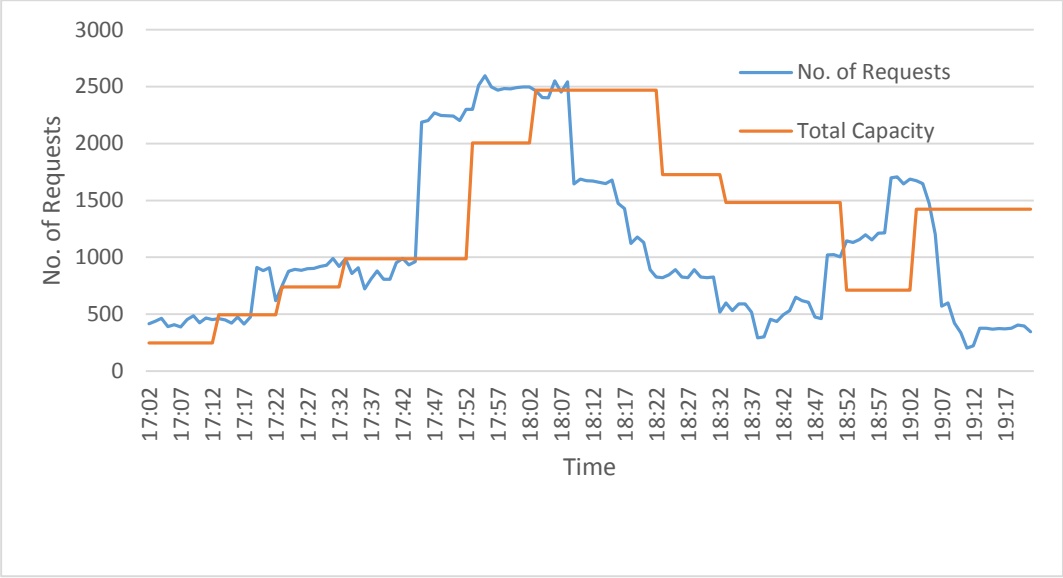

Figure 12 Auto-scaling of Instances to Adjust predicted load

Total number of instances that were launched by the VM management agent at different times during the experiment to handle the dynamic changes in workload are given in table 10. Table shows that the number of instances currently being used vary with the changes in the workload. As soon as an increase in workload is predicted, new instances are made available to incorporate the incoming load before it actually hits the application. It improves the performance of the application remarkably as instances take time to launch which often results in poor performance or even unavailability of the application. Our multi-agent system keep some instances in buffer in ready state so they are made available as soon as the demand changes without any delay.

Table 9 Total number of requests and instances for each time window

Time Windows	No. of requests	No. of instances
1	2312	1
2	4341	1
3	5856	2
4	8672	3
5	8655	4
6	18483	4
7	24627	7
8	22316	8
9	13883	8
10	8400	5
11	4835	4
12	6883	4
13	13262	4
14	9810	4
15	3610	4

4.5 Conclusion

In this chapter we have presented the results of a series of experiments that were carried out to evaluate the proposed multi-agent system. The first section describes the results of the prediction techniques. Three machine learning algorithms Linear Regression, SMO Regression and Multilayer Perceptron were used for forecasting web application traffic. The time window used is of 10 minutes. The results show that out of three linear regression performs better than the rest followed by SMO regression and Multilayer Perceptron.

Overall performance of the machine learning algorithms can be improved by changing the size of the history data to be used and the future time interval for forecasting. The small the interval the better is the accuracy of the techniques. Also, the accuracy of the techniques is affected by the sudden changes in workload, when change is sudden and large the accuracy decreases a little bit before it improves again.

The surge detection techniques have shown very high accuracy. If looks for the predictions results and total capacity of the current running instances and makes a decision if a surge is coming or not and results show that the techniques works very well in case of multiple types of surges depending upon their size. The last section represented the results of the auto-scaling agent. Auto-scaling agent plans out the management of the surge using predicted workload, current capacity of resources and surge status. The performance of the auto-scaling technique is compared with any traditional auto-scaling technique which works based on the workload prediction. The results show a very significance improve in performance which is due to the use of buffer instances. As soon as an increase in workload is predicted, new instances are made available to incorporate the incoming load before it actually hits the application. It improves the performance of the application remarkably as instances take time to launch which often results in poor performance or even unavailability of the application. Our multi-agent system keep some instances in buffer in ready state so they are made available as soon as the demand changes without any delay.

Chapter IV

5. Conclusion and Future Work

This chapter concludes the research work carried out, experimental results and the future directions.

5.1 Conclusion

Cloud computing has become a powerful computing platform for web applications because of the ability to auto-scale based on the demand from the users. The current auto-scaling mechanisms seem promising when the application workload follows a certain pattern, but they do not perform well when a sudden surge in traffic hits the web application. In order to address this problem we formulized a hypothesis that sudden surges in web traffic applications can be predicted using web site's log, and managed intelligently using agents. We formulated a number of research questions described in chapter 1 to prove this hypothesis true. To find the answer to the first question, what a web log can tell about usage pattern of a web application, we used the machine learning techniques Linear Regression, Multilayer perceptron and SMO regression. These techniques make use of the web site log to predict the future traffic. Experimental results have shown that we can successfully predict the future workload of web application with accuracy. The second question is answered by implementing a multi-agent system that consists of a number of agents each responsible for an important task. The agents can communicate with each other making it easy to share information among them and it also reduced the need of human intervention to a minimal level as all tasks are carried by agents and they makes decisions and take actions autonomously. The answer to last question is found by implementing instance buffer to handle the sudden surges in web application traffic and evaluation of this approach compliments the answer. Through careful experimentation and analysis of the answers of all these research questions we conclude that the hypothesis is true. Sudden surges in web application traffic can be detected and multi-agent can be used to handle these surges intelligently. We developed a multi-agent framework in JADE for auto-scaling cloud infrastructure automatically to meet the workload requirement of applications

for sudden surges in workload. Three machine learning techniques were implemented for workload forecasting and evaluated their results for real time data in cloud environment. The results show that linear regression work best for prediction, followed by SMO regression and multilayer perceptron.

Auto-scaling is an integral part of the system. Scaling algorithm computes the scaling plan for predicted workload and instances are made available before the actual arrival of the workload which prevents the system from slowing down due to exhausting of resources. In order to maintain the performance level during a sudden surge an instance buffer approach which contains instances in ready state is used. However there is a tradeoff between the cost and performance of the application during surges. If buffer size is kept small it reduces the overall cost of the application but decreases performances as compared to large buffer where enough instances are present to handle the peak load.

5.2 Future work

In future we will evaluate our framework with more workload patterns. We will evaluate our system for different types of web applications which vary in nature and the types of instances they need. We will explore other measures that can be useful to increase the performance such as reducing the bandwidth consumption by disabling dynamic content of the web application. We will also look into other attributes of application which can be used as an indicator of an incoming surge such as the link structure of the web site. We are also planning to work on prediction techniques to improve the accuracy of prediction results.

References

[1] Rajan S. S. (2010) "Dynamic scaling and elasticity-windows azure vs amazon ec2," http://dotnet.syscon.com/node/1626508. Online: Accessed 04 Sept 2013

[2] Varia, J. (2008) "Amazon white paper on cloud architectures," http://aws.amazon.com/whitepapers/. Online: Accessed 04 Sept 2013

[3] Reese G. (2008), "On why I don't like auto-scaling in the cloud," http://broadcast.oreilly.com/2008/12/why-i-dontlike-cloud-auto-scaling.html. Online: Accessed 09 Dec 2013

[4] Jiang, J., Lu, J., Zhang, G., & Long, G. (2013, May). Optimal cloud resource auto-scaling for web applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on* (pp. 58-65). IEEE.

[5] "Amazon Web Services." [Online]. Available: http://aws.amazon.com/ Online: Accessed 27 Aug 2014

[6] Chieu, T. C., Mohindra, A., Karve, A., & Segal, A. (2009, October). Dynamic scaling of web applications in a virtualized cloud computing environment. In *e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on* (pp. 281-286). IEEE.

[7] Meiländer, D., Ploss, A., Glinka, F., & Gorlatch, S. (2012, January). A dynamic resource management system for real-time online applications on clouds. In *Euro-Par 2011: Parallel Processing Workshops* (pp. 149-158). Springer Berlin Heidelberg.

[8] Moreno-Vozmediano, R., Montero, R. S., & Llorente, I. M. (2009, June). Elastic management of cluster-based services in the cloud. In Proceedings of the 1st workshop on Automated control for datacenters and clouds (pp. 19-24). ACM.

[9] Miguel-Alonso, T. L. B. J., & Lozano, J. A. (2013) Comparison of Auto-scaling Techniques for Cloud Environments.

[10] Maurer, M., Brandic, I., & Sakellariou, R. (2011). Enacting SLAs in clouds using rules. In Euro-Par 2011 Parallel Processing (pp. 455-466). Springer Berlin Heidelberg.

[11] Dutreilh, X., Rivierre, N., Moreau, A., Malenfant, J., & Truck, I. (2010, July). From data center resource allocation to control theory and back. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on (pp. 410-417). IEEE.

[12] Han, R., Guo, L., Ghanem, M. M., & Guo, Y. (2012, May). Lightweight resource scaling for cloud applications. In Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on (pp. 644-651). IEEE.

[13] Hasan, M. Z., Magana, E., Clemm, A., Tucker, L., & Gudreddi, S. L. D. (2012, April). Integrated and autonomic cloud resource scaling. In Network Operations and Management Symposium (NOMS), 2012 IEEE (pp. 1327-1334). IEEE.

[14] Right Scale Cloud Management. http://www.rightscale.com/ Online: Accessed 27 Aug 2014

[15] Microsoft Windows Azure. https://www.windowsazure.com/en-us/ Online: Accessed 12 Sept 2012

[16] Gong, Z., Gu, X., & Wilkes, J. (2010, October). Press: Predictive elastic resource scaling for cloud systems. In Network and Service Management (CNSM), 2010 International Conference on (pp. 9-16). IEEE.

[17] Lim, H. C., Babu, S., Chase, J. S., & Parekh, S. S. (2009, June). Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds* (pp. 13-18). ACM.

[18] Park, S. M., & Humphrey, M. (2009, May). Self-tuning virtual machines for predictable escience. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid* (pp. 356-363). IEEE Computer Society.

[19] Huang, J., Li, C., & Yu, J. (2012, April). Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics*,

Communications and Networks (CECNet), 2012 2nd International Conference on (pp. 2056-2060). IEEE.

[20] Bankole, A. A., & Ajila, S. A. (2013, March). Cloud client prediction models for cloud resource provisioning in a multitier web application environment. In *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on* (pp. 156-161). IEEE.

[21] Iqbal, W., Dailey, M. N., Carrera, D., & Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, *27*(6), 871-879.

[22] Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., & Yuan, L. (2010, July). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on* (pp. 514-521). IEEE.

[23] Kupferman, J., Silverman, J., Jara, P., and Browne, J. (2009) Scaling into the cloud. Technical report, University of California, Santa Barbara; CS270 - Advanced Operating Systems, http://cs.ucsb.edu/jkupferman/docs/ScalingIntoTheClouds.pdf Online: Accessed 20 Jan 2015

[24] Roy, N., Dubey, A., & Gokhale, A. (2011, July). Efficient autoscaling in the cloud using predictive models for workload forecasting. In Cloud Computing (CLOUD), 2011 IEEE International Conference on (pp. 500-507). IEEE

[25] Iqbal, W., Dailey, M. N., & Carrera, D. (2011, December). Black-box approach to capacity identification for multi-tier applications hosted on virtualized platforms. In *Cloud and Service Computing (CSC), 2011 International Conference on* (pp. 111-117). IEEE.

[26] Baryshnikov, Y., Coffman, E., Pierre, G., Rubenstein, D., Squillante, M., & Yimwadsana, T. (2005, September). Predictability of web-server traffic congestion. In *Web Content Caching and Distribution, 2005. WCW 2005. 10th International Workshop on* (pp. 97-103). IEEE.

[27] Saripalli, P., Kiran, G. V. R., Shankar, R. R., Narware, H., & Bindal, N. (2011, December). Load prediction and hot spot detection models for autonomic cloud computing. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on* (pp. 397-402). IEEE.

[28] Napagoda, C. (2013). Web Site Visit Forecasting Using Data Mining Techniques. *International Journal Of Scientific & Technology Research, 2*(12), 170-174.

[29] Dalmazo, B. L., Vilela, J. P., & Curado, M. (2014, August). Online Traffic Prediction in the Cloud: A Dynamic Window Approach. In *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on* (pp. 9-14). IEEE.

[30] Jung, S., Kim, C., & Chung, Y. (2006). A prediction method of network traffic using time series models. In *Computational Science and Its Applications-ICCSA 2006* (pp. 234-243). Springer Berlin Heidelberg.

[31] Cortez, P., Rio, M., Rocha, M., & Sousa, P. (2006, July). Internet traffic forecasting using neural networks. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on* (pp. 2635-2642). IEEE.

[32] Amari, S. I., & Wu, S. (1999). Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, *12*(6), 783-789.

[33] Seber, G. A., & Lee, A. J. (2012). *Linear regression analysis* (Vol. 936). John Wiley & Sons.

[34] MediaWiki. http://www.mediawiki.org/wiki/MediaWiki. Online: Accessed 04 Jun 2015

[35] *WikiBench: A Web hosting benchmark. http://www.wikibench.eu. Online: Accessed 04 Jun 2015*

[36] *Wikipedia access traces. http://www.wikibench.eu/?page_id=60. Online: Accessed 04 Jun 2015*