RESOURCE PROVISIONING FOR MULTI-TIER WEB APPLICATIONS BASED ON WORKLOAD PREDICTION IN CLOUD COMPUTING

A

Thesis

Submitted to

For the award of

DOCTOR OF PHILOSOPHY (Ph.D.)

in

COMPUTER SCIENCE AND ENGINEERING

By

Parminder Singh

41400088

***S*upervised By :**

Dr. Pooja Gupta

***C*o-Supervised By :**

Dr. Kiran Jyoti

LOVELY FACULTY OF TECHNOLOGY AND SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

2019

DECLARATION

This thesis is an account of research undertaken between August 2014 and March 2019 at The Department of Computer Science and Engineering, Lovely Professional University, Phagwara, India.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Parminder Singh
Registration no. 41400088
Department of Computer Science and Engineering
Lovely Professional University, Phagwara, India

CERTIFICATE

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. He has submitted the Ph.D. thesis **Resource Provisioning for Multi-tier Web Applications Based on Workload Prediction in Cloud Computing** under my guidance and supervision. The present work is the result of his original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Ph.D. thesis is fit for the submission and fulfillment of the conditions for the award of Ph.D. degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Dr. Pooja Gupta
Associate Professor
Department of CSE
Lovely Professional University,
Phagwara, India

Dr. Kiran Jyoti
Assistant Professor
Department of IT
Guru Nanak Dev Engineering
College, Ludhiana, India

ABSTRACT

Cloud computing is the on-demand delivery of computing power and storage capacity on a pay-as-you-go system. The elasticity feature enables the application providers to provision and de-provision the resources in an autonomic manner as per the real-time demand of the applications.

Cloud resources are often under-provisioned or over-provisioned due to the dynamic workload of web applications. The reactive auto-scaling technique delays to launch Virtual Machines (VMs) instances due to instantiation time. A prediction model is a key to design the proactive auto-scaling approach, which prepares the resources before the arrival of actual workload. However, it is a challenging task for application providers to gain cost efficiency, high availability and optimal resource utilization under the flash crowd. The fundamental challenges need to be addressed are prediction model with high accuracy, dynamic resource provisioning mechanism with efficient resource estimation and selection of VMs during the scaling decision.

In this thesis, the current state-of-the-art is enhanced with time series prediction model, auto-scaling technique and resource provisioning mechanism. A Technocrat ARIMA and SVR Model (TASM) has been developed with a combination of statistical and machine learning models. A classification approach has been designed to identify workload features and select the prediction model at the initial stage. The model parameters are selected through residual testing. Further, Robust Hybrid Auto-Scaler (RHAS) has been developed as per Monitor-Analyze-Plan-Execute (MAPE) architecture for autonomous computing. The monitoring phase collects the user requests and

infrastructure level parameters. Analysis and planning algorithms have been designed with a queuing model and threshold-based rules to decide the scaling action from CPU utilization, response time and time series prediction. Furthermore, Triangulation Resource Provisioning (TRP) technique has been developed with a profit-aware surplus VM selection policy for the execution phase. This policy aims to balance cost and load trade-off while selecting the VM to scale-in from the in-service VMs capacity.

TASM prediction model demonstrates a significant improvement in prediction accuracy of web applications workload. The proposed time-series prediction model is efficient to forecast seasonal and non-seasonal workload patterns. RHAS is a robust approach to estimate the number of VMs and takes the scaling decision for multi-tier web applications in cloud data center. The hybridization of reactive and proactive auto-scaling methods achieves cost efficiency and high availability. TRP resource provisioning mechanism with profit-aware surplus VM selection policy plays a vital role in scale-in decisions.

The experiment results show a significant improvement in total cost, response time, scaling overhead and consistency in CPU utilization as compared to existing techniques. It also mitigates the short-term under-provisioning in cloud data centers due to the flash crowd. The proposed resource provisioning mechanism is a profitable approach for web application providers and ensures the Quality of Service (QoS) to the end-users.

ACKNOWLEDGEMENTS

I take this opportunity to express my sincere thanks to everyone who has helped me in various capacities to carried out this research and prepare the report.

I am delighted to thank our respected supervisors Dr. Pooja Gupta and Dr. Kiran Jyoti who have offered tremendous support in the completion of this research. Their unparalleled knowledge, judgment, and moral fiber were together with their expertise.

I acknowledge the Department of Computer Science and Engineering, Lovely Professional University to provide me the appropriate resources and financial support to pursue the doctoral degree. I am grateful to the administration staff at the Centre for Research Degree Programmes for the numerous applications.

I would also thank my parents, wife, daughter, friends, and contemporaries for their co-operation and compliance. I cannot cherish a greater fortune other than having them in my life. Their care and love are indispensable for my achievements.

Parminder Singh

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACF	Auto Correlation Function
AP	Application Profiling
AR	Autoregression
ARIMA	Autoregressive Integrated Moving Average
ASP	Application Service Provider
CSP	Cloud Service Provider
CTH	Control Theory
FR	Fuzzy Rules
IaaS	Infrastructure as a Service
LR	Linear Regression
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAPE-K	Monitor-Analyze-Plan-Execute-Knowledge
ML	Machine Learning
MSE	Mean Square Error
PaaS	Platform as a Service
PACF	Partial Auto Correlation Function
QoE	Quality of Experience
QoS	Quality of Service
QTH	Queuing Theory

RHAS	Robust Hybrid Auto Scaler
RIL	Reinforcement Learning
RMSE	Root Mean Square Error
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service Level Objectives
SVM	Support Vector Machine
SVR	Support Vector Regression
TASM	Technocrat ARIMA and SVR Model
TR	Threshold-based Rules
TRP	Triangulation Resource Provisioning
TSA	Time Series Analysis
VM	Virtual Machine
WWW	World Wide Web

CHAPTER 1

INTRODUCTION

Cloud computing provides the computer resources as utility. The virtualization of resource becomes robust with technologies such as grid and cloud computing [8]. The cloud providers creates the large IT infrastructure and rent the resource as pay-per-use model [9]. The application providers are moving their application to cloud for cost benefit [10]. It helps the organization to concentrate on the main business line instead of focusing on technologies experiments. The cloud users can access their applications anywhere in the world. Fundamental of cloud computing is to deliver storage, softwares and computing as a service. National Institute of Standards and Technology (NIST)[11] define cloud as : "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models."

At the abstract level there are three types of cloud services.

1. **Infrastructure as a Service**

   The service providers enable the users to provision CPU, RAM, disk, network and other computing resources where the customer can deploy their applications and softwares. The cloud infrastructure is not managed by the users. The consumers

have the control over its applications.

2. **Platform as a Service**

   The consumer can deploy and manage the application onto the cloud. Infrastructure can be created by user or given by third party. It can be created via programming languages or tools.

3. **Software as a Service**

   The facility provided to users to use its application running on cloud through certain interface such as browser. The user is able to do the configuration of its deployed application, other resources are managed by cloud providers.

As a computing platform, the application providers host different applications in cloud environment such as web applications, scientific workflows and data analytical services. In traditional manner, the web applications are hosted on rented servers or in private infrastructure. These mechanisms does not support scalability features, which results in over-provisioned and under-provisioned conditions. In cloud, elasticity feature enrich the users to grow/shrink the resources from unlimited virtual resources. Thus, the cloud becomes the ideal environment to host web applications which have dynamic nature in workload. The application providers are interested to move their web applications from traditional infrastructures to cloud environment.

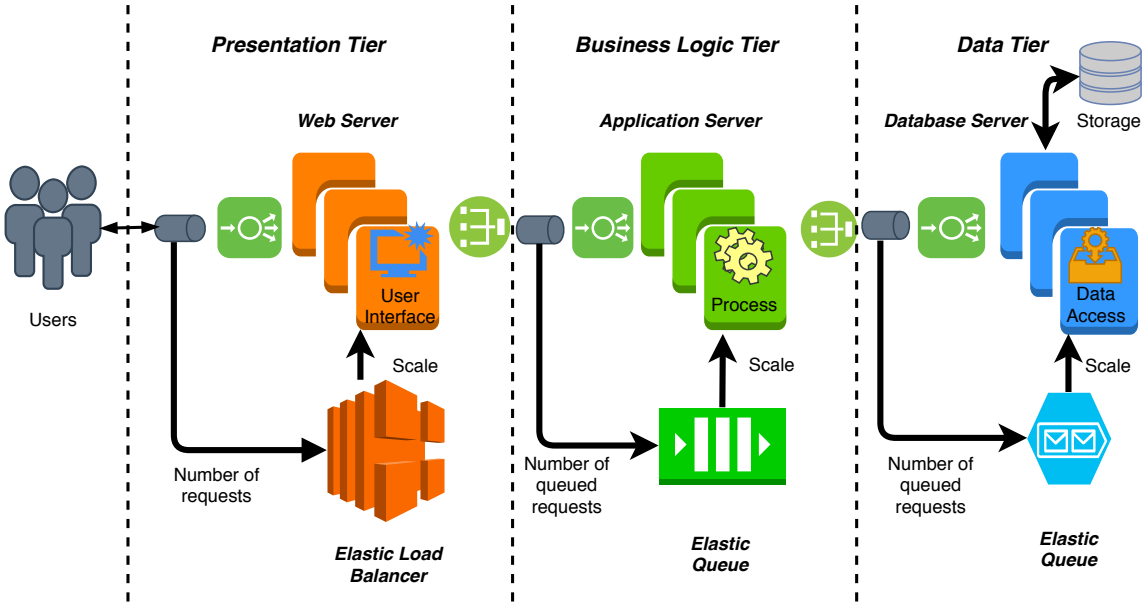

Figure 1.1: Three-tier cloud application [1]

Presentation tier, business logic tier and data tier is implemented by web server, application server and database server shown in Figure 1.1.

Functionalities of Web Server

- Serve the static web content and accept/denying the client request
- Request passing to application server
- Response receiving and delivered back to client

Functionalities of Application Server

- Request receiving from web server
- Fetch data from database and process the business logic
- Processed data send back to Web server

Functionalities of Database Server

- Store the data in structured manner
- Store the data in unstructured manner
- Replication management

Earlier single tier architecture was used dedicated server for each tier such as web server (Load Balancer), application server and database server. This architecture is not suitable according to the elasticity feature of cloud computing. Most of the companies are now using multi-tier architecture. Each tier in application serves a specific purpose. The important benefit of multi-tier architecture is to manage the elasticity features. Resource management of multi-tier web applications are still a challenge due to higher interdependency between the different layers [2].

1.1 Challenges in Web Applications Resource Management

To manage the web applications in cloud environment, we consider three major aspects: 1) Workload prediction, 2) Auto-Scaling, and 3) Provisioning. The workload prediction

is a process to take historical data and predict the incoming workload. Furthermore, the auto-scaler planned the scaling decisions based on workload prediction or resource utilization. The provisioning finally executes the auto-scaling decision to scale up/down or do-nothing the resources in cloud environment by considering hardware and application provider constraints.

1.1.1 Challenges in Workload Prediction

Cloud application workload patterns are constantly changing. The predictive workload model is a key to autonomous cloud resources scaling. This technique have the potential to for cost reduction and make effective use of resources. The workload for web applications is usually combined for various pattern over time. The single predictive model can not predict various types of patterns in the workload.

1.1.2 Challenges in Auto-scaling

The scalability enriches the application provider to dynamically provision the computing power and storage capacity on-demand for their applications. The carefully designed auto-scaling approach can reduce the cost for the application provider while maintaining the Quality of Service (QoS). It is crucial to understand when to scale up and down the resources. The state-of-art algorithms mostly focus either on reactive or proactive technique.

1.1.3 Challenges in Resource Provisioning

The dynamic resource provisioning arranges the resources on-demand according to the application workload. The over-utilization and under-utilization of resources can be prevented with autonomic resource provisioning. In the literature, the Service Level Agreement (SLA) aware, load-aware, resource-aware and user behavior aware solutions have proposed. The solutions are rigid for a particular metrics which provides benefit either to end users or to the application providers. The scaling decision is taken based on the planning phase of auto-scaling module. Further, this decision need to revisit based on the infrastructure level constraints set by the application providers such as maximum number of on-demand VMs to be provision for a particular application. The technique required to give better QoS within the budget of application provider.

1.2 Research Issues and Objectives

The main objective of this thesis is to explore the prediction models, auto-scaling techniques and resource provisioning mechanisms for web applications in cloud infrastructure.

- **Workload Analyzer** - It is responsible for the forecasting of future interest of the applications. The forecast workload series sent to load predictor and performance modeler modules.
- **Performance Modeler** - It selects the VMs to be distributed by taking into account the anticipated interest from the workload analyzer module. It also monitors the execution of running VMs. Performance modeler module estimate the required number of VMs and pass the estimate to application provisioner.
- **Application Provisioner** : It gets the user request from the admission controller and pass it to the VMs in IaaS. It is responsible for monitoring the performance of VMs. This module requests the resource provisioner for scale up/down the VMs instances based on analysis and planning phase [4].

To explore the above mentioned problems, first we need to develop the workload predictor that can give better accuracy for dynamic user requests. The following issues need to be addressed:

- How to convert historical web logs to discrete model?
- How to classify the incoming workload?
- How to select the prediction model with highest accuracy?
- How to check the accuracy of predicted workload?

After workload prediction, we have to use the predicted workload to estimate the number of resources in next scaling interval to minimize the renting cost, this need to tackle following issues:

- How to monitor the user level and infrastructure level parameters?

- How to analyze the resource utilization?
- How to estimate the number of resource in next interval?
- How to minimize the resource oscillation?
- How to observe and handle the SLA violation?

Thirdly, provisioning should ensure the required resources availability for web applications. To ensure the profit for application providers following issues need to be taken care:

- How to execute the planning policy to scale up/down/nothing as per application providers constraints?
- Which VM to choose in scale down decision?
- How to provision the resources under different pricing policy?
- How to ensure profit to the application providers with QoS to the users?

Therefore, the researcher summarize the following objectives.

1. To propose workload prediction model on different time series for Multi-tier web applications.
2. To develop an auto scaling approach for efficient resource selection for application servers in multi-tier cloud environment.
3. To develop profit-aware dynamic resource provisioning algorithm for the virtualized cloud environment.
4. To evaluate and analyze the proposed model with existing models on different QoS parameters e.g. cost, response time, reliability, elasticity etc.

1.3 Research Methodology

The expected outcome of the research work is to develop a robust resource provisioning mechanism based on workload prediction for multi-tier web applications to provide the QoS to user with minimum SLA violation. A flow chart in Figure 1.2 is a list activities

to achieve the mentioned objectives followed by the research methodology for each objective.

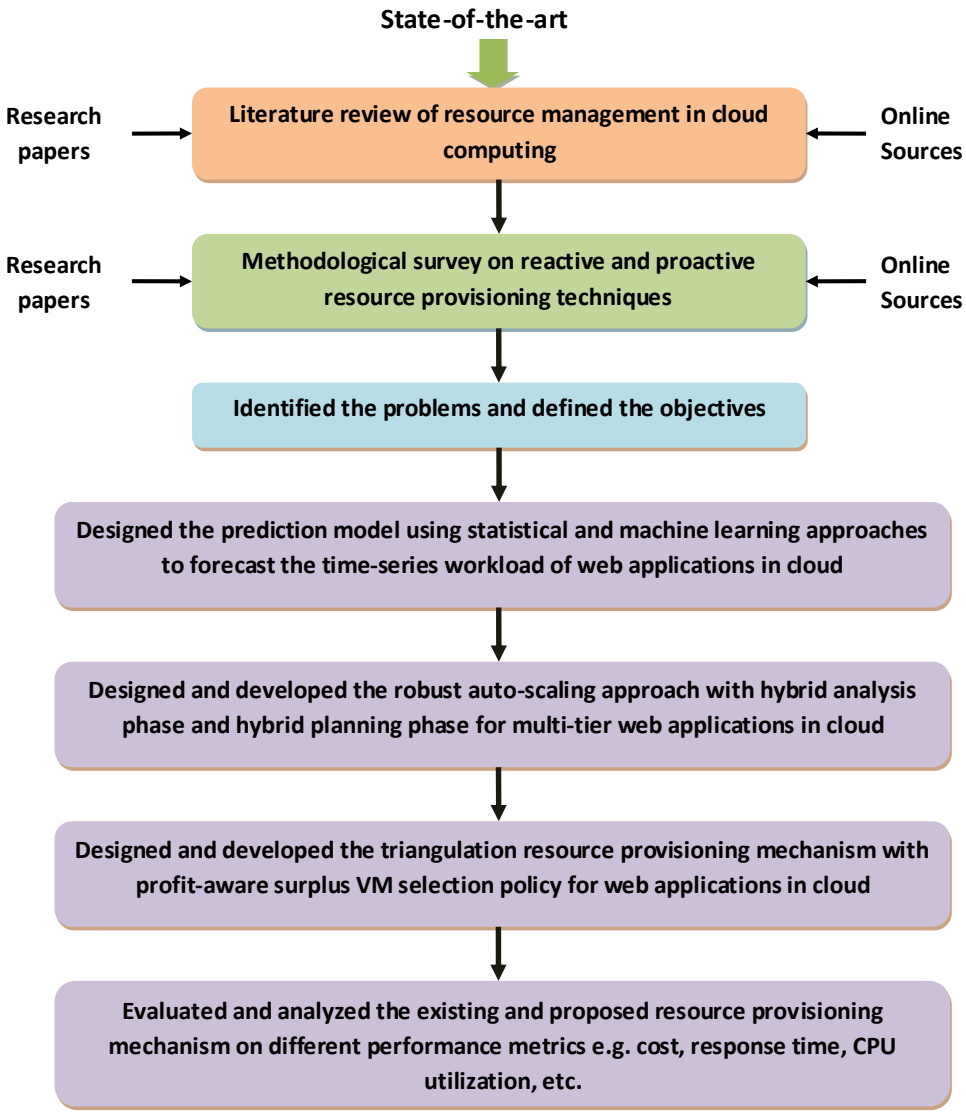

Figure 1.2: Research methodology flowchart

1. To achieve the first objective, the statistical and machine learning techniques has applied. The LR, ARIMA and SVR model has been selected for predicting time series based on classification technique for the workload of multi-tier web applications. The real-time web server logs of NASA and ClarkNet has used for experimental evaluation. The proposed model has developed and analyzed in R-tool. The queuing model used to calculate the accuracy of the proposed model in the resource provisioning of the cloud environment.

2. To achieve the second objective, a robust hybrid auto-scaling approach has been presented. The analysis phase and planning phase in the MAPE-K loop designed with the combination of a reactive and proactive approach. The scale up/down decisions are planned by considering the time series prediction in the first objective and current system state. The simulation performed with Cloudsim toolkit by incorporating auto-scaling libraries.

3. To achieve the third objective, a triangulation resource provisioning mechanism designed with workload prediction, response time and CPU utilization. A profit-aware surplus VM selection policy designed by considering the cost and load on VM in the current scaling interval. The simple and professional executor used to simulate the proposed algorithm. The simulation performed on Cloudsim toolkit with executer libraries.

4. Reliability of predictive resource provisioning model with different models has been evaluated by analyzing the execution of multi-tier web application on the developed environment on different QoS matrices. e.g. RMSE, MAPE, cost, response time, CPU utilization, etc.

1.4 Thesis Contribution

To give the answer of defined research questions, the thesis contribution is mentioned as per following:

- A taxonomy and literature survey of auto-scaling and resource provisioning techniques for web application in cloud.

    1. A detail investigation has done to study various existing resource provisioning techniques in cloud computing.
    2. The mentioned techniques classification has been done as per the common characteristics.
    3. Future research direction in the area of auto-scaling is presented.

- A technocrat ARIMA and SVR model to predict the web applications dynamic workload to gain highest accuracy in short term prediction.

1. The technocrat workload prediction cloud architecture.
2. A workload characterization and classification technique.
3. A technocrat ARIMA and SVR model for time series forecasting of dynamic workload.

- A robust auto-scaler for web application in cloud with highest resource utilization and minimum cost.
  1. A MAPE-K based cloud architecture for web applications.
  2. The design and development of hybrid analysis approach for resource auto-scaling for Web applications in the cloud.
  3. The design and development of a hybrid planning approach for scaling decisions in cloud infrastructure.

- A profit-aware triangulation resource provisioning mechanism for application service providers.
  1. A TRP resource provisioning architecture for web applications.
  2. The resource provisioning mechanism is designed for ASPs for web applications in the cloud.
  3. The surplus VM selection algorithm is developed with the profit-aware approach.

1.5 Thesis Organization

The structure of thesis and its dependencies are shown in Figure 1.3. Chapter 2 is related to literature survey and taxonomy of auto-scaling and provisioning techniques. Chapter 3 is focused on the workload prediction of web application in cloud computing. Chapter 4 is presented the auto-scaling techniques for web application management. Chapter 5 is presented the profit-aware resource provisioning of web application in cloud computing. In detail, the organization of thesis as follows:

- Chapter 2 presents the methodological survey and taxonomy on resource auto-scaling and provisioning techniques for web application in cloud computing. This chapter is partially derived from:

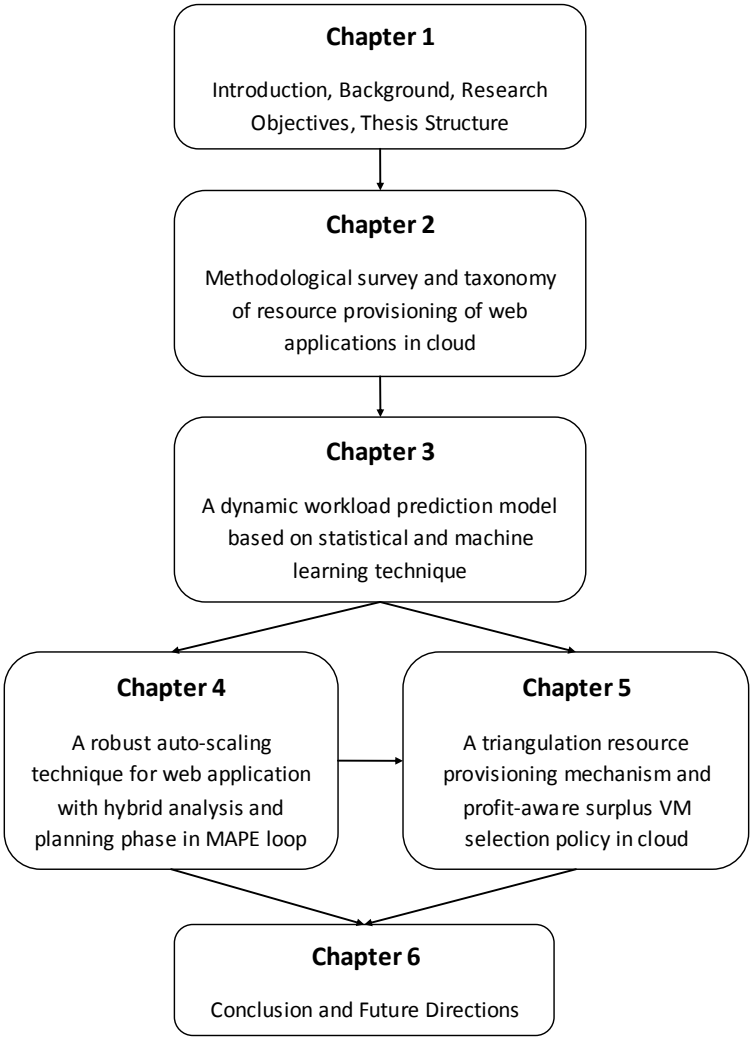

Figure 1.3: Chapter wise thesis organization

  - **Parminder Singh,** Pooja Gupta, Kiran Jyoti, Anand N., "Research on Auto-Scaling of Web Applications in Cloud: Survey, Trends and Future Directions", *Scalable Computing: Practice and Experience* (Accepted), 2019. (Scopus, ESCI)

- Chapter 3 describes the proposed a technocrat ARIMA and SVR model for dynamic workload prediction of web applications. This chapter is derived from:

  - **Parminder Singh,** Pooja Gupta and Kiran Jyoti, "TASM: Technocrat ARIMA and SVR Model for Workload Prediction of Web Applications in Cloud", *Cluster Computing, Springer* (Published), 2018. (Scopus, SCIE 1.6 IF)

- Chapter 4 explains the devised robust auto-scaling technique with hybrid analysis

and planning algorithms in cloud architecture for application service providers. This chapter is derived from:

  - **Parminder Singh,** Pooja Gupta and Kiran Jyoti, "A Robust Auto-Scaling for Web Applications in Cloud Computing", *Cluster Computing, Springer* (Under Review), 2018. (Scopus, SCIE 1.6 IF)

- Chapter 5 presents the triangulation resource provisioning technique and profit-aware surplus VM selection in scale-down decision in the execution phase of cloud infrastructure. This chapter is derived from:

  - **Parminder Singh,** Pooja Gupta and Kiran Jyoti, "Triangulation Resource Provisioning for Web Applications in Cloud Computing: A Profit-Aware Approach", *Scalable Computing: Practice and Experience* (Accepted), 2019. (Scopus, ESCI)

- Chapter 6 present the conclusion of thesis finding and introduce the possible future directions. This chapter is partially derived from:

  - **Parminder Singh,** Pooja Gupta, Kiran Jyoti and Anand N., "Research on Auto-Scaling of Web Applications in Cloud: Survey, Trends and Future Directions", *Scalable Computing: Practice and Experience* (Accepted), 2019. (Scopus, ESCI)

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Cloud computing is an emerging technology, which provides processing, bandwidth, and storage as a service. Elasticity is the main characteristic of cloud computing, at runtime resources are allocated and de-allocated from the processes as per increment or decrement in the requirement. The resource pools seems unlimited for the users and can acquire or release the resources anytime [12]. The regular monitoring of given services is required to ensure the Quality of Service (QoS) and also need to fulfill the Service Level Agreements (SLAs). The SLA violation leads to the penalty to cloud providers. It is a big challenge for the service providers to provide services within the budget and raise profit from datacenters. The QoS assures the behavior of cloud services towards reliability, availability, elasticity, cost, time, etc. [13]. The infrastructure providers offer different pricing policies, companies such as Amazon [14] provides resources for a fixed price per hour. Thus, the providers should decide the resources for the processes while maintaining the SLAs terms. In auto-scaling, decision-making techniques to allocate the number of resources to different processes are categorized as reactive and proactive. The reactive technique regularly watches events such as CPU, workload, queue, etc., and perform the elastic operation on the resources as per threshold rules. Proactive forecasting methods are used to predict the traffic from the past workload. So far none of the technique is splendid in all the cases [15].

The reasons for the ambiguity and diffusion in resource allocation for cloud environment are heterogeneity of resources, dynamic application requirements, and failures. As for now, none of the technique can tackle aforesaid issues. Autonomic cloud computing can provide the self-optimization, self-protecting, self-healing, and self-configuration scaling [16].

2.2 Origin of Autonomic Computing

The word autonomic comes from the Biological system. The human body is working in a self-managing manner without the interference of human. Autonomic Nerves System (ANS) can handle problems like uncertain conditions, dynamic behavior and fault handling. The similar way an automatic system works on the basis of human guidance and behave accordingly in case of environment change like software and hardware requirements change or in case of fault occurrence. ANS handle the human body's functions like sweating to keep cool, pupil adjustment due to sunlight, digestion, and breathing, the autonomic system handles the functionality of computing system and application deployed without human intervention [17].

2.2.1 Advancement in Autonomic Computing

Researchers have started building the autonomous and self-managing system due to the reason for heterogeneity, complexity, failures, and increase of knowledge in the computing system [18]. Dynamic Assembly for Systems' Adaptability, Dependability, and Assurance (DASADA) is a project of the Department of Defense (DoD) developed to handle the problem of a distributed system. Author working on 19 technologies given by the DOD department and working on the adaptability and dependence to ensure high-performance [19].

IBM took many initiatives for autonomic computing. IBM introduced an autonomic computing system to manage failures, risk, and resources. Author invited the researchers to work on the autonomic system using technologies like adaptive algorithms, ANS, artificial intelligence, feedback control like technologies make it possible to develop autonomic computing system for Grid computing [20]. IBM mentioned four features of self-management as self (configuration-optimization-healing-protection). IBM

is a leader organization to develop autonomic computing system, but still a grand challenge to develop a computing system that can do self-management. The manual and automatic system are not able to manage the heterogeneous computing environment [21].

Prasher and Harri [22] developed an autonomic grid computing system in 2005. The robust autonomic computing system proposed for heterogeneous components. The model also provided the features like dynamic resources, grid application management, and deployment. Autonomic cloud computing system was first introduced by kim et al. [23] in 2009. Ensemble Kalman Filters (EnKF) is used to investigate the workflow history and works on autonomic objectives (acceleration, resilience, and conservation). CometCloud has been used for resource management techniques for Hybrid infrastructure. Traditional methods are not appropriate to fulfill the QoS requirement in autonomic cloud computing. In this field, many areas are open to research.

2.3 Auto Scaling

Auto-scaling is a technique to dynamically adjusts the resources allocated to elastic applications as per the incoming workloads. Auto-scalers in the cloud environment are mostly generic while some are application specific to meet the SLA, QoS and cost requirements. The challenge in auto-scaling for the web applications is to dynamically grow or shrink the resources to meet fluctuated workload requirement. Autonomous scaling techniques work without human intervention. Autonomic systems are self (configuring-optimizing-protecting-healing) [23]. The auto-scaling following the MAPE loop: Monitoring (M), Analysis (A), Planning (P) and Execution (E) [24] shown in Figure 2.1.

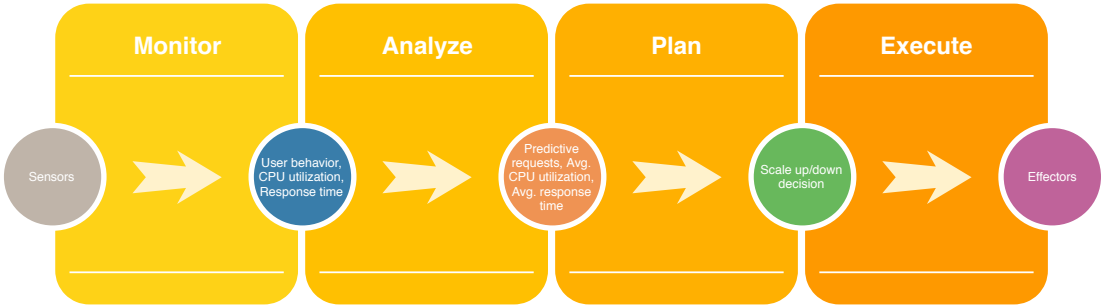

Figure 2.1: MAPE loop in resource provisioning

- **Monitoring**: The monitoring system collects the information from a cloud environment about the compliance of user expectations, resource status, and SLA violation. It provides the state of infrastructure to the cloud provider, and users get to know about application status with expected SLA. Auto-scaling protocols are decided on the basis of performance metrics for web applications. Ghanbari et al. [25] suggested parameters such as resize numbers, operating interval, decision duration, decision threshold, refractory period and instance bounds. Generally, metrics provided by cloud providers are related to VM management; otherwise, it could be taken from the operating system. The proxy metrics are used to reduce the complexity of metrics such as hypervisor level and application level (e.g., CPU utilization, workload).

- **Analysis**: The collected information is further processed in the analysis phase. It gathers all information from metrics, current system utilization and prediction information from historical workload. Some auto-scaler are working on a reactive approach. The decision is taken after analyzing the current system state. The threshold values are fixed to scale in/out decisions, while others are using a reactive approach or both. Reactive is a sophisticated approach because there is always a delay between the settings of resources for scaling decision. The VM startup time varies from 350 to 400 seconds [26]. Flash crowd and events are still a challenge with the reactive approach.

- **Planning**: Analysis phase evaluates the present state, now the planning phase has to decide to scale up/down or scale in/out to compliance with SLA and profit trade-off.

- **Execution** : Execution phase is already decided in the planning phase. Cloud providers API is responsible for the execution of planning. The client is unaware of the issues in the execution phase. VMs are available to users for a certain period, the startup time of VM takes some time, and these delays have been already discussed with the user in resource SLA.

2.3.1 Types of Auto-scaling

There are two types of auto-scaling techniques.

1. **Reactive**

   The reactive approach makes the dynamic decision at the runtime of the application. Most of the models followed the threshold approach to create a new VM. When the CPU utilization reaches a certain threshold, scale up and down decisions are made [27]. The approach is simple and easy to implement, but it is relatively slow and does not consider the VM boot-up time.

2. **Proactive**

   The VM startup time cannot be neglected. Therefore, the proactive approach forecasts from the historical workload pattern or CPU utilization. The short time history is used to forecast future workload [28]. It considers the startup time in advance. Challenge in proactive approach is the accuracy of prediction models.

2.3.2 Types of Auto-scaling Policies

There are two types of auto-scaling policies.

1. **Vertical Scaling**

   It provides the scale up/down approach and system can change the machine configuration at run time. Such as CPU cores, ram, a disk can change. The RackSpace [29] is providing the facility to change the machine configuration. Most of the service provider does not support this feature [15].

2. **Horizontal Scaling**

   It provides the Scale in/out approach. The system can create and destroy the VM. All service providers provide these features.

Resource provisioning mechanisms have the following research challenges:

Under-provisioning: The application has not sufficient resources to serve all incoming requests from the servers. It may happen due to flash crowd or events, or poor auto-scaling algorithm. This situation leads to SLA violation and application providers have to pay the penalty to the users, and reliability of the providers also get affected. The servers takes some time to be back in the normal state. This duration is know as cool down period. The auto-scaling policy need to arrange enough resources as per the application requirements which ensure the QoS to the end-users.

Over-provisioning: In this situation, the number of resources to process the application is more than the required resources. Service provider's reliability is increased in this case. SLAs violation is minimum in over provisioning and up to a certain level beneficial to handle the fluctuated workload. It affects the profit of the service provider, and on-demand services become costly for the clients. There is no perfect solution exists either in automatic or autonomous scaling.

Oscillation: It is a pack of both unwanted situations. Rapid resources scaling is taking place without considering the effect on application performance. The condition can be avoided using static and dynamic threshold value fixed for the VMs scaling. A cooling down period is another approach used to handle the oscillation [30].

2.4 Elastic Applications

The elastic applications are dynamic in nature towards change in workload and variables. The load balancer is responsible for the management of elastic applications. Cloud computing applications are managed on VMs, and VMs are managed by the servers. Auto-scaler is a decision maker for the scaling of resources. The system is said to be autonomous because human intervention is not required. Many cloud applications (e.g., Video streaming, web applications) are elastic in nature [31]. Most of the articles about elastic applications considered the web applications as compared to other elastic applications. Web applications consist of three tiers: Presentation, Application, and Database. Most of the articles focused on the application or business tier scaling. The major issue with auto-scaling is to scale the small running jobs, while long-running jobs are coming under the scheduling problem.

2.4.1 Web Applications Architectures

Earlier single tier architecture has used the dedicated server for each tier such as a web server (load balancer), application server and database server. The load balancer transfers the load among other instances of single layer architecture. This architecture is not suitable according to the elasticity and scalability features of cloud computing.

Most of the companies are now using multi-tier architecture (e.g., Amazon), where each tier in architecture serves a specific purpose as shown in Figure 2.2. The most

important benefit of multi-tier architecture is to manage the scalability and elasticity features. Resource management of multi-tier applications is still a challenge due to higher interdependencies between the different layers [2]. Web tier, application tier, and database tier are deployed on the web server, application server, and database server. The responsibilities of the servers are:

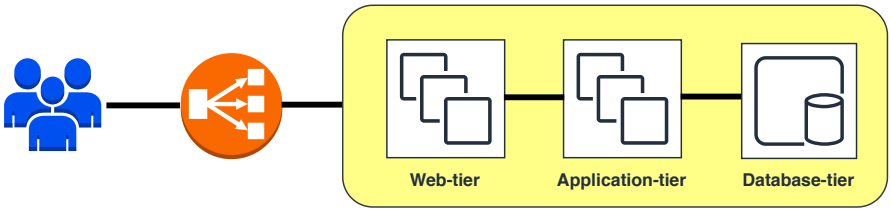

Figure 2.2: 3-tier Web architecture [2]

- **Web Server:** It accepts or rejects the incoming client request. Another important work of web server is to serve the static content. It also passes the request to the application server. The response is delivered back to the user.

- **Application Server:** It receives the request from the web server. The literature survey is biased towards the scaling of VMs for the application server. Business logic is processed by this tier. It requests the database for the required data. Optimization of queries can be done at this level also. After processing data again send back to the web servers in the desired format.

- **Database Server:** Database management system is working at this level. The application server connected this tier via database connectivity APIs (e.g. JDBC, ODBC). Single tenant and multi-tenant databases compatible with the cloud architecture (e.g. Oracle 12c). Structured and unstructured relational database management system is used to store the data and pass the required data to application servers.

Another type of applications is service-based web applications such as Facebook and e-commerce website of Amazon. Each service represented as the node is connected with another service through directed edges and whole system abstracted as a directed graph. The application further classified in Micro-services and Service-Oriented Architecture (SOA). The each tier in the the web application is required separate VM in cloud environment. The present study is mainly focused on multi-tier web applications.

2.4.2 Application Benchmarks

Server performance and scalability can be checked with benchmarking applications. It is the combination of web application and synthetic session based request to the application. Some of the benchmarking applications are.

- RUBiS [32] Auction website model similar to eBay.com. It provides useful features (browsing, bidding, selling) and backings three sorts of client sessions: guest, purchaser, and dealer. The applications comprise of three primary parts: Apache load balancer server, JBoss application server, and MySQL database server. Benchmark updated in 2008.

- TPC-W [33] It is a transaction processing and database benchmark. Its a complex shopping site for an online book store. It provides feature browsing, ordering, and shopping. The performance is a meter on a different time scale. It was declared an absolute benchmark in 2005.

- CloudStone [34] It is develop by Rad group, University of Berkley. It gives the facility to measure the performance of multi-language and multi-platform tool for web 2.0 application.

  Other least used benchmarks are SpecWeb [35], TPC-C [36] and RUBBoS [37] working in similar manner. It is best to use these benchmarking to measure the performance in Real or simulated environment.

2.5 Capacity Management of Web Applications

As the growing demand for information services demand three things- scalability, availability and cost optimization from the service providers. Scalability means the service provider can fulfill the demand of unpredictable customer's crowd with minimum performance degradation. Availability means the user can access the website anytime and anywhere. The service provider should provide a predictable response time as per SLA. Figure 2.3 represents the capacity planning model and its working.

The first step is to understand the architecture of an application in the business models and measurable goals module. In the second step, it characterizes the workload as per different sites. The session describes the use of the site by different users e.g. use of

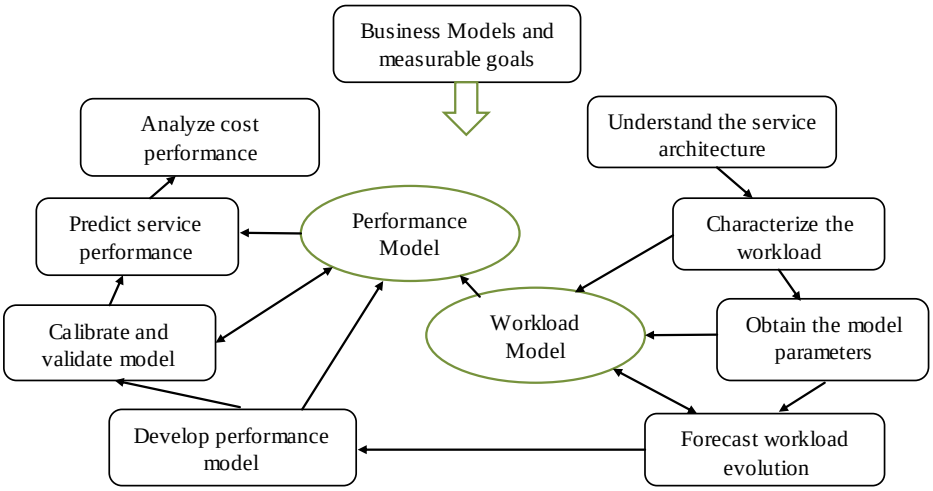

Figure 2.3: Capacity planning process [3]

shopping website by the user opens multiple pages at the same time to browse different items. The third step is to get model parameters from monitoring and measure the performance of web services. Next step is to forecast the future workload. Further conduct performance modeling for different workload pattern. Performance metrics (response time, throughput and resource utilization) should map the actual requirement. Service performance prediction should be done to predict the performance of the system. At last, it performs the cost-benefit analysis and, plan the corrective actions in order to meet the business objectives [3].

2.6 QoS-aware Cloud Computing

The agreement between customer and cloud service provider are based on the functional requirements like time, price and, the non-functional QoS parameters such as reliability and availability. Many techniques are there to resolve the expense and time constraints. However, the existing techniques are unable to fulfill the QoS requirements. The rise of Big data and IoT technologies required robust resource management techniques to efficiently use the resources. Cloud computing gives new sights to the popular wireless sensor network field to overcome the limitations of battery life and storage capacity [38]. Garg et al. [39] proposed the architecture having features like dynamic resource scaling and configuration. Heterogeneous workload considered without taking consideration of application behavior.

2.7 Cloud Provisioning Architecture

The adaptive provisioning approach designed for dynamic workload, uncertain behavior and resource estimation error [4]. The architecture is shown in Figure 2.4. The service providers administered various components in the architecture. The SaaS layer accepted the requests for application instances. The accepted user requests are sent to the PaaS layer. The present study implements in the PaaS layer.

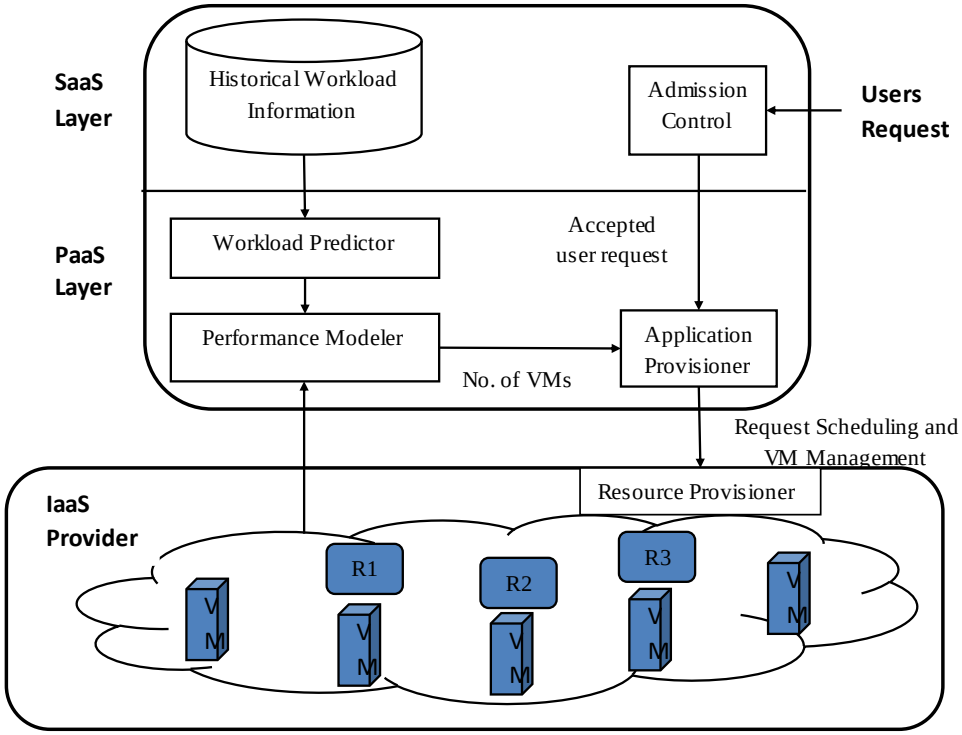

Figure 2.4: Cloud provisioning architecture [4]

1. **Workload Analyzer** : It is responsible for the prediction of future incoming workload for the cloud applications. The forecast value inputs to the load predictor and performance modeler module.

2. **Load Predictor and Performance Modeler** : It selects the VMs to be distributed and taking into account the anticipated requests from workload analyzer module. It also monitors the execution of running VMs. Queuing model is used to predict the required VMs from the arrival rate of request that can meet the QoS measurements.

3. **Application Provisioner** : It gets the user request from the admission controller

and passes it to VMs in IaaS. It is responsible for monitoring the performance of VMs. Performance modeler module estimates the required number of VMs and passes the estimate to application provisioner. The analysis and planning phase takes the scaling decision. Further, the decision inputs to the resource provisioner [4].

2.8 Resource Demand Estimation in Cloud

Once the predicted workload is achieved from the workload analyzer, the next step is to resource demand estimation within the cloud. The resource estimation cannot decide merely on the basis of workload, its the combination of workload and the current state of a system. The queuing method is widely used in the industry to estimate the resources [40] as shown in Figure 2.5. Resource demand estimation model are webPropht[41] and same has extended for cloud is CloudPropht [42].

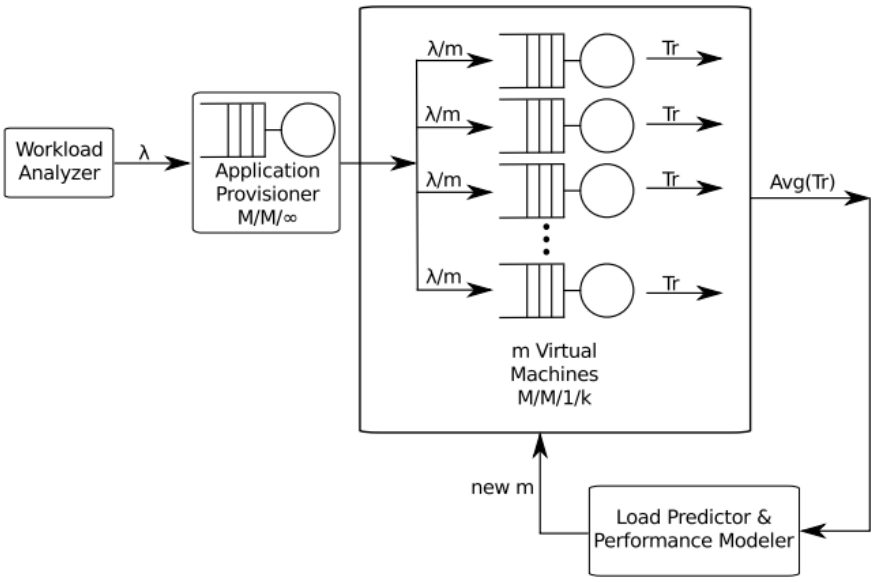

Figure 2.5: Queuing model for cloud data center [5]

Application provisioner gets the request from the admission controller working on the M/M/1/K queue model. K is a queue size and defined at the time of SLA negotiation of service time (Ts) and single request execution time (Tr). Admission control system rejects the request if VM exceeds K thus it will not forward the request to application provisioner. The accuracy of the resource demand prediction depends on workload prediction, so the resource demand is a variable. Various techniques for demand prediction

are Response time approximation, Service Demand Law, Linear regression, Kalman filter, Optimization, Machine learning, Maximum likelihood estimation, Gibbs sampling [43].

2.9 Taxonomy of Auto-scaling

Auto-scaling feature automates the capacity up and down process in a cloud environment. The user can specify the condition for scale up and down in interface developed by cloud provider like Amazon EC2 [44]. The vendor RightScale [45] helps the user to set the rules for auto-scaling. The user defines an upper and lower threshold rules to instantiate scaling. The rule-based approach assumes that consumer has much knowledge of the domain, but in actual condition, most of the users are not aware of scaling rules. Apart from queuing theory, threshold technique, reinforcement learning, time series analysis, and control theory techniques are used in auto-scaling [31]. The main purpose of auto-scaling is job scheduling, SLA monitoring and run time scaling.

The taxonomy of auto-scaling is considered the following aspects:

1. Auto-scaling technique
2. Proactive, reactive or mixed (R/P)
3. Horizontal, vertical scaling or both (H/V)
4. Performance metrics (e.g. input request rate, CPU load time)
5. SLA parameters (e.g. response time)
6. Pricing model (e.g. On-demand, reserve, spot instance)
7. Monitoring tools
8. Experiment setup (e.g. Simulator, testbed or real provider)
9. Workload (e.g. Real or synthetic)
10. Approach (e.g. cost-aware, load-aware, etc.)

The researcher in the cloud computing used various techniques for analysis and planning phase in MAPE loop to automate the scaling process. In literature, widely

used techniques classified in 7 major approaches such as threshold rules, fuzzy rules, application profiling, machine learning, queuing theory, control theory and time series analysis.

Auto-scaler is a crucial component in cloud computing. Auto-scalers are grouped into two categories: Reactive and Proactive. The reactive approaches took the scaling decision by analyzing the current state of the system. Proactive technique analyzes the historical data and take scaling decision. Most of the techniques used hybridization of these methods. So it's challenging to classify the methods on two broad groups.

Cloud service providers widely use horizontal scaling as elasticity feature. Cloud providers are providing a fix and customizable resources where the user can configure VMs by specifying memory, cores, bandwidth, etc. Some articles developed the auto-scaling techniques using vertical scaling, where the user can re-configure the VMs resources such as CPU, memory and network bandwidth as per the requirement change. The Centurylink service provider gives the service to scale the CPU cores without vertical downtime.

Performance metrics is an essential tool to enhance the reliability of the users moving their applications to the cloud. The articles in literature used various metrics to check the performance of their model such as SLA, CPU load time, request rate, etc.

The service providers need to know the expectations of the customers. It is the master service agreement between the customer and service providers related to various performance outage issues. This document describes the responsibilities of the customer and service providers. It helps the customer to compare the performance among different service providers. The commonly consider performance metrics in SLA related to auto-scaling are response time, budget, cost, throughput, etc.

Cloud providers are categories the cloud resources in three different pricing models: on-demand, reserved and spot-instances. On-demand resources gives performance guarantee to the target application. The number of resources grows or shrink as per the workload change through elasticity feature of cloud computing. The reserved resources are a fixed number of resources. Amazon provides the spot instances relatively cheaper than the on-demand resources. Spare capacity instances sell through auction mechanism and user acquire the resources by submitting the bid. Single cloud and multi-cloud resource estimation challenges are different for multi-tier web applications.

Monitoring tools act as performance indicators. It helps to determine the scaling decisions. Monitoring interval defines the performance of auto-scaler. Balanced performance can be achieved by selecting the right monitoring interval according to the application. Amazon cloudwatch monitoring used by many articles whether some are using custom monitoring tool.

Experiment characteristics considered for the implementation of the model has been reviewed in the taxonomy of auto-scaling. Synthetic and real workload used by an article for the analysis of the model. Cloudsim simulator used by many articles, while some manuscripts used real testbed for the study of the model.

The newly added feature in the taxonomy is the approach of the authors on above of the techniques to auto-scale. In the literature, the vision of investigators is to improve one or more factors such as cost, resource optimization, QoS, SLA violation, etc. Further, the researcher get clear idea about the articles which improve the specific objectives.

2.10 Survey on Auto-scaling Techniques

This literature survey is focusing on auto-scaling and resource provisioning techniques specifically for web applications in cloud computing. Researcher performed a survey on the auto-scaling technique which specifically focused on analysis and planning phase. Auto-scaling of cloud computing has a large number of articles published. The researcher put the papers in the associated category by identifying the approach, algorithms used in the research paper for a better understanding. The researcher again revised the models, metrics, monitoring tools, etc., from the articles and create tables according to the classification of techniques. The symbol '-' in tables represent the lack of information in the article.

Earlier surveys have been conducted by many authors [46, 47, 31, 48, 49], but the regular updates in cloud infrastructure and persistent research yielding the new research areas. There is a need to explore the present challenges and new research area in this field. This survey augments the existing studies and recent research articles to describe the research challenges for the web applications in cloud computing. Guitart et al. [46] have done a survey of Internet applications deployed on dedicated or shared data centers. It focuses on the web server's admission control issue. This service is related to

the SaaS layer of the cloud architecture. The elastic nature of web applications is more relevant with auto-scaling and can be done in two ways: horizontal or vertical. Identification has done on the basis of approach reactive or proactive. The metric used by the technique is identified in the survey. Workload used in the analysis of the approach is also discussed along with the experimental setups. SLA parameter has to check the validity of the model, so SLA parameters are collected from the papers.

Resources are the major factor in auto-scaling. To the best of our knowledge, no existing survey has identified all types of resources used in a cloud environment. Existing surveys considered only on-demand resources in cloud infrastructure. In this study, the researcher has identified other type of resources (e.g. Spot-instances, reserve and on-demand) for application tier elastic layer.

Another important survey done by Manvi and Shyam [47] on resource management in the cloud, but little attention has given to auto-scaling. Lorido-Botran et al. [31] focused on the elastic application including web application. The article classifies the technique and very useful literature survey, but the survey did not focus on the type of resources such as on-demand, reserve and spot instances.

In this study, the researcher carried out a very diverse survey by including these surveys with specifically to web applications auto-scaling techniques. A survey has done on the basis of auto-scaling technique. Lorido-Botran et al. [31] has used the classification technique and the investigator have added more approaches along with mentioned former groups. This section covers the literature survey on auto-scaling techniques. Each article is reviewed on the basis of auto-scaling taxonomy. The classification of auto-scaling techniques are:

1. Application Profiling (AP)
2. Control Theory (CTH)
3. Fuzzy Rules (FR)
4. Machine Learning (ML)
5. Queuing Theory (QTH)
6. Threshold-based Rules (TR)
7. Time Series Analysis (TSA)

2.10.1 Application Profiling

It is a method to find the critical point of utilization of resources while running of application workload (real or synthetic). The methods of application profiling are shown in Figure 2.6. This technique is classified into two categories:

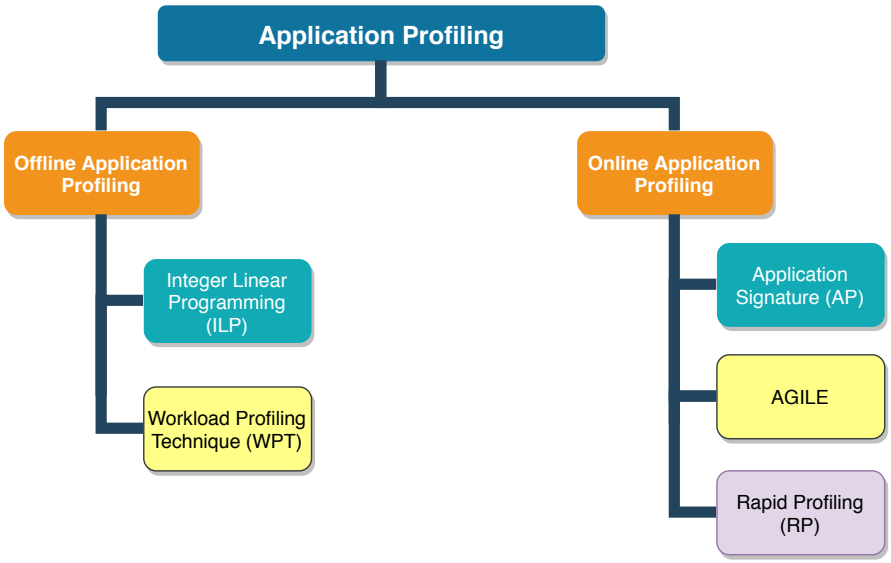

Figure 2.6: Methods of application profiling

1. **Offline Application Profiling:** This method of profiling is performed in a detailed manner in resource provisioning of tasks at different stages of workload. The advantage of offline profiling is to perform profiling manually after the applications are updated. The different methods of offline application profiling are:

   (a) **Integer Linear Programming (ILP):** It is a mathematical optimization problem. In this partial or all variables are integer. The constraints and objective function is linear [50].

   (b) **Workload Profiling Technique (WPT):** This technique collects the workload information. The workload modeling performed to estimate the performance under the stressed conditions. The workload profiling done with various types of testing techniques such as limit finding, soak testing, etc. [51, 52, 53].

2. **Online Application Profiling:** This method of profiling is dynamic in nature and fulfill the needs of the fine grained tasks that immediately require working virtual

machines with different workloads. The different techniques of online application profiling are:

(a) **Application Signatures (AS):** This technique identifies a small set of classes of workload and classifies them according to workloads. The resource allocation of workload is cached at runtime [54].

(b) **Elastic Distributed Resource Scaling (AGILE):** In this technique, wavelets are used for fulfilling the resource demand prediction with much large time for the applications servers to start up before falling short of performance [55].

(c) **Rapid Profiling (RP):** In this technique, the individual performance of the virtual machine instance is profiled after it is obtained from the cloud. This technique helps to judge the best tier for the profiled instance of virtual machine [56].

Table 2.1 shows the taxonomy of reviewed articles in this section.

Table 2.1: Taxonomy on application profiling based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[50]	AP + ILP	P	H	Arrival rate	Response time and cost	Reserve	-	Custom testbed	Cost-Aware
[52]	AP	R/P	H	Arrival rate, No. of Servers	Response time	on-demand	Custom tool	Custom testbed. 38 Intel Xeon servers.	Capacity Management
[51]	AP + TSA	P	H	Arrival rate, CPU usage	Throughput	on-demand	Custom tool. 5 minutes.	Custom testbed. MediaWiki application.	Workload and Customer aware
[53]	AP	R	H	CPU load	Response time and availability	on-demand, spot instances	Simulated.	Custom simulator	Fault tolerant

Continued on next page

Table 2.1 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[54]	AP + CMAC	P	V	CPU, I/O, memory, swap	Response time	on-demand	Custom tool (Script)	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)	Workload Aware
[55]	AP	P	H	Number of user requests, number of VMs, response time	Response time, cost	on-demand	-	Custom testbed. Olio application + Custom decision agent (VirtRL)	Cost-Aware/ Resource-Aware
[56]	AP + ANN	P	V	CPU and memory usage	Response time	on-demand	Custom tool	Custom testbed. Xen + 3 applications (TPC-W, TPC-C, SpecWeb)	Load /Capacity Aware

2.10.2 Control Theory

This method is used in the analysis and planning of the MAPE loop as shown in Figure 2.7. It automates the processing systems such as data centers, storage systems, and web server systems. The main goal is to automate the scaling process. The controller maintains the controlled variable y and manipulated variable y_{ref} matches to the desired level. Manipulated variable act as an input to the target system, the output is measured by the sensor.

The different categories of control systems are:

- **Open-loop:** It is also called the non-feedback method. In this method, feedback is not considered for validating the desired goal. The output is calculated using the present state of the system.
- **Closed-loop:** The closed loop systems are further categorized into two categories:

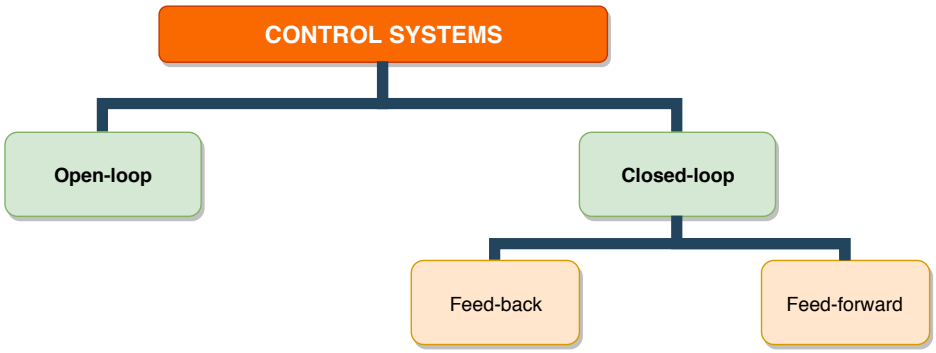

Figure 2.7: Categories of control system

- **Feed-back:** The monitoring of the output and deviation from the goal is monitored by the feedback controller. The working of the feedback system is shown in Figure 2.8.
- **Feed-forward:** The anticipation of any kind of error in output is performed by the feed forward method. The action is performed after considering the type of error in the system.

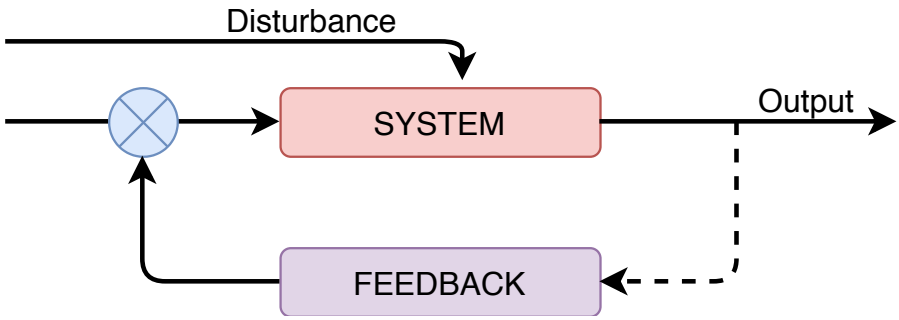

Figure 2.8: Working of feedback control system

The feedback controller is mostly used in the reviewed articles. It further divided into several categories [57] is shown in Figure 2.9:

- **Fixed Gain Controllers**: It is the simplest of all the available controllers. The different types of fixed gain controllers are Proportional-Integral-Derivative (PID), Proportional-Integral (PI) and Integral (I). PID is the most common controller.

$$u_k = K_p e_k + K_i \sum_{j=1}^{k} e_j + K_d \left(e_k - e_{k-1}\right) \tag{2.1}$$

Manipulated variable is denoted by u_k (e.g. New number of VM); e_k is the difference between set point y_{ref} and y_k is the output; k_i, k_p and k_d are the integral,

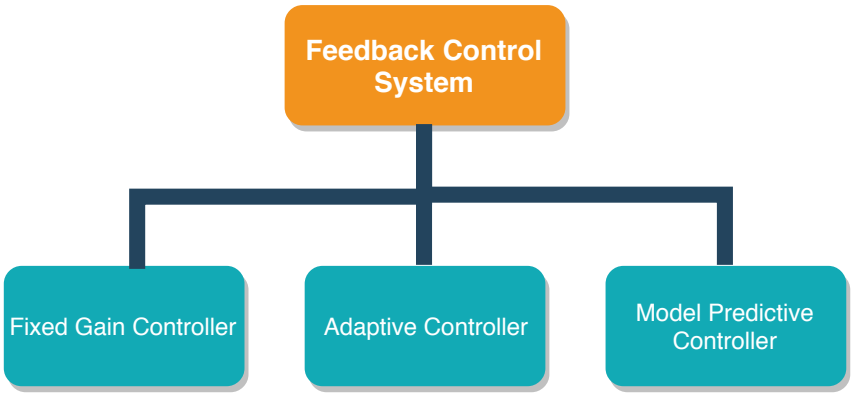

Figure 2.9: Categories of feedback control system

proportional and derivative gain parameters, which further adjust as per the target system.

Lim et al. [58, 59] applied the I controller to manage the required VMs on average CPU utilization. Park and Humphrey [60] deployed PI controller to adjust the resources for batch jobs and used their execution process. k_p and k_i (gain factors) adjusted manually according to trail-and-error [58] basis or target application model. Zhu and Agrawal [61] adjusted a PID controller derivative term using the RIL agent. The agent learns to reduced the sum of squared error of control variables without affecting budget and time constraints over time.

- **Adaptive Controllers**: These controllers are the adaptive controllers that work as per the online data made available by target systems. It is incapable of handling the flash load. It is further classified into three categories:
  - Self-tuning PID Controller (SPID)
  - Self-tuning Regulator (STR)
  - Gain Scheduling (GS)

The adaptive controller is also used in the literature. For example, Ali-Eldin et al. [62] used the two models, adaptive and proactive controllers for scaling-down, based on the input workload and dynamic gain parameters. Scaling-up is done using a reactive approach. Author also devised proportional controller using proactive technique [63]. Padala et al. [64] introduced an adaptive PID controller for MIMO and used second-order ARMA for non-linear relationships

between performance and resource allocation. The controller can fit the disk I/O usage and CPU. Bodik et al. [65] used smoothing splines with gain. A gain-scheduling adaptive controller applied to estimate the number of servers for input workload. Kalyvianaki et al. [66] combined the Kalman filter with controllers (SISO and MIMO) for CPU allocation to VMs. Gambi and Toffetti [67] predicted job completion using kriging model. The master node enqueued all the incoming request.

- **Model Predictive Controllers (MPC)**: It is proactive in nature that considers the present output and also forecasts the future behavior of the system. One of its categories is Look-ahead controller [68]. An optimization problem is solved by considering the cost function. These types of controllers comes under the proactive approach. Roy et al. [68] devised workload forecasting model using the ARMA model and look-ahead controller. Fuzzy model predictive controller developed using the fuzzy model [69].

  Farokhi et al. [70] devised the auto-scaling technique for vertical memory. The application performance and resource utilization considered in developing the hybrid controller. The objective is to consume less memory for the task, and achieve the highest memory utilization. The author achieved 83% memory utilization. This work can further extend to increase memory utilization and considered the cache memory in vertical scaling.

  An auto-scaling task is highly dependent on the design of the controller and the target application. The goal of the controller is to add and remove the resources in the auto-scaling process. The problem still persists, when the workload is dynamic and non-linear. General auto-scaling model is required, that can be an adaptive controller with MPCs.

As discussed earlier, the controller has to tune the input variable (number of VMs) to calculate the output variable (CPU load). In order to achieve this goal, a model has been devised to represent the formal relationship, which determines how input parameters affect the output variables. This relationship is known as a transfer function in control theory. PID controller represents with a linear equation, there is also the possibility to define with a non-linear equation. Simple PID controller Single-Input-Single-

Output (SISO) is used, but there is also a provision to use Multiple-Input-Multiple-Output(MIMO). Following performance models have been used in literature:

- ARMA(X) [64]: ARMA (Auto-regressive Moving Average) model is used to define the characteristics of time series and draw future predictions. ARMAX (ARMA with eXogenous input) devised the relationship between two-time series.
- Kalman Filter [66]: It is used to make a prediction of time series. It is a recursive algorithm.
- Smoothing Splines [65]: It is a polynomial function used to smooth the curves to avoid the noisy observations.
- Kriging Model or Gaussian Process Regression [67]: Statistical framework combined with linear regression to predict future values. Unsampled data have been used to perform the forecasting with a confidence measure.
- Fuzzy Model [71, 69, 72]: These models are used with fuzzy rules. Set of membership elements assigned a degree of value between 0 and 1 (Boolean logic).

Table 2.2 shows the taxonomy of reviewed articles in this section.

Table 2.2: Taxonomy on control theory based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[60]	CTH: PI controller	R	V	Job progress	Job deadline	on-demand	Sensor library	Custom testbed, Applications + HyperV	Predictive /Cost-aware
[58]	CTH: PI controller + ES	R	H	CPU load, request rate	-	on-demand	Hyperic HQ (Xen)	Custom testbed. Xen + ORCA + Web service	-
[64]	CTH: MIMO adaptive controller + ARMA	P	V	CPU usage, disk I/O, response time	Response time	on-demand	Xen + custom tool. 20 seconds	Custom testbed. Xen + (RUBiS, TPC-W, media server)	SLO based

Continued on next page

Table 2.2 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[65]	CTH: Gain-scheduler + Smooth-ing splines + Liner Re-gression	P	H	No. of requests and servers, response time	Response time	on-demand	20 seconds	Real provider. Amazon EC2 + CloudStone benchmark	Resource-aware
[66]	CTH: SISO and MIMO con-trollers + Kalman filter	P	V	CPU load	Response time	on-demand	Custom tool. 5 – 10 seconds	Custom testbed. Xen + RUBiS application	Resource-aware
[59]	CTH: PI controller (Propor-tional thresh-olding)	R	H	CPU load, request rate	-	on-demand	Hyperic SIGAR. 10 seconds	Custom testbed. Xen + Modified cloudStone (using Hadoop Distributed File System)	SLO based/ Work-load based
[62]	CTH: Adaptive con-trollers + QTH	R/P	H	Number of re-quests, service rate	Number of re-quests not handled	on-demand	Simulated	Custom simu-lator in Python	SLA Based
[63]	CTH: Adaptive, Propor-tional controller + QTH	R/P	H	Number of re-quests, requests in buffer, service rate	-	on-demand	Simulated. 1 minute	Custom simu-lator in Python	Load Aware

Continued on next page

Table 2.2 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[67]	CTH: Self-adaptive controller + Kriging model	P	H	Number of incoming and enqueued requests, no. of VMs	Execution time	on-demand	-	Custom testbed. Private cloud + Sun Grid Engine (SGE)	QoS aware
[73]	CTH: Fuzzy controller + TS	P	H	Number of hits, RMSE	Response time	on-demand	Custom tool (Fuzzy controller)	Custom testbed. Azure Small VM	Workload Aware
[74]	CTH: Learning automata	P	H	Scaling overhead	Response time	on-demand	Custom tool. 5 minutes	Cloudsim	SLA Aware
[70]	CTH: Hybrid Controller + TS	P	V	CPU and memory utilization	Response time	on-demand	Control interval using / proc/ meminfo	Custom testbed. Xen hypervisor (RUBBoS application)	App. based/ Resource based

2.10.3 Fuzzy Rules

One of the rule-based technique for the auto-scaling approach is fuzzy rules. In this technique, rules are defined using if-else conditions. The major advantage of fuzzy based auto-scaler over the threshold based rule-based approach is linguistic terms e.g. low, medium, high. It uses the control system as a performance model for estimating the required resources (output variable) for the workload (input variable). Fuzzy sets are formed from the input and output variables, the membership function defines this process between the $(0, 1)$ interval. Fuzzification is the process of transforming input variables into fuzzy sets. The inverse transformation of single numeric values to best inferred fuzzy value is known as defuzzification.

The fuzzy rule-based model used to construct the auto-scaler. This auto-scaler is

not able to handle the dynamic workload because most of the components of the fuzzy controller are fixed at the design time (e.g. Rule set, membership function). Regular update in the fuzzy controller with on-line monitoring can make the system of adaptive nature. It could better handle the dynamic workload [69, 72]. Xu et al. [72] proposed a model to apply the fuzzy-controller at the application tier, and forecast the resources required input workload. Want et al. [69] used the same method for the database tier. Predict the required resources r_{t+1} for the time step $t+1$, considering no flash workload during that time step. Grimaldi et al. [75] developed a dynamic approach for horizontal scale-out for the dynamic workload. Author proposed a fitness function: $\mathscr{F}_k = \sum_{i}^{N} a_i \frac{m_{i,k}}{\mathscr{R}_i}$. Here, metric observation represented by $m_{i,k}$ and $\mathscr{R}_i$ at k sample time $a_i \varepsilon [0,1]$. The model works well under the failure of VMs, and robust for different workloads.

Many research articles extended the work with the neural fuzzy controller. Four-layer neural network used to represent the fuzzy model [71]. The first layer belongs to the input node. The second layer represents each input variable membership to the fuzzy set. Layer three determines the precondition part of fuzzy rules. Final layer act as a defuzzifier, which used to convert the layer 3 in numeric output. The rules and membership of nodes are formed on-line with the help of parameters and structure learning. Table 2.3 shows the taxonomy of reviewed articles in this section.

Table 2.3: Taxonomy on fuzzy rules based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[72]	FR	R	H	CPU allocation, Arrival rate	Through-put	Reserve con-tainers	Custom tool	Custom testbed + applications (Java Pet Store)	Workload aware / SLA aware
[76]	FR	P	H	No. of servers per tier	End-to-end delay	on-demand	-	Simulation	QoS aware
[69]	FR	P	V	No. of queries, I/O band-width, CPU load	Response time, through-put	on-demand	Xentop. 10 seconds	Custom testbed, RUBiS and TPC-H applications + Xen	Qos aware

Continued on next page

Table 2.3 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[71]	FR + ANN	P	V	Number of requests, resource usage	End-to-end delay	on-demand	Custom tool. 3 minutes	Simulation	Workload aware
[77]	FR	R	H	Arrival rate	Response time	on-demand	Custom tool	Custom Simulation	SLA aware
[78]	FR	P	H	Number of hits, RMSE	SLO	on-demand	Custom tool.	Microsoft Azure Cloud	Load aware
[75]	FR + PID controller	P	H	CPU load and network usage	CPU usage(%), NETIN, NETOUT	on-demand	Amazon cloud-watch	Setup on Amazon EC2	QoS aware

2.10.4 Machine Learning

Machine learning is a technique that is used online learning for constructing dynamic approach for the estimation of resources. It is a self-adaptive technique as per the workload pattern available. The machine learning algorithms reclassified into various categories. The review is made for the different categories used in this particular literature.

- **Reinforcement Learning (RIL)**: It is one of the widely used machine learning approaches. Its agent performs an action as per the received environmental state. As in a cloud environment, auto-scaler acts as an agent. An agent works on the principle of trial and error method. It gets a response or reward for each action performed [79]. The optimal scaling decision is taken by sensing the current state, performance metric, and type of application. The auto-scalar or agent works to yield more rewards. The main objective of machine learning is to control the data. The purpose of the agent is to fetch appropriate action of each states.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{0}^{\infty} \gamma^k r_{t+k+1} \tag{2.2}$$

At time $t+1$ the rewards gained are R_{t+1}, the γ factor is the discount factor. The

value function $Q(s,a)$ known as Q-value function defines the policy. The $Q(s,a)$ values evaluates the cumulative rewards for each state s by execution action a.

$$\mathscr{Q}(s,a) = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1|S_t} = s, a_t = a\right\} \tag{2.3}$$

This proactive learning method makes the decision with the state of application about the future reward (e.g. response time). The result is generated after the complete execution of the application on the cloud. The phases analyze and plan of MAPE process are covered by RIL techniques. Firstly, data about the application and rewards are taken from the lookup table (or any other structure) for later use (analyze phase). In the planning phase, the data is used to take the scaling action.

To apply RIL to auto-scaling, some basic elements need to define: first, the action set A, the state space S, and the reward function R. The first two depend upon the type of scaling: horizontal and vertical. Reward function depends upon the cost to acquire the resources (VMs, network bandwidth, etc.), and SLA violation penalty cost. While considering the horizontal scaling, the state defines as input workload and the number of VMs.

Tesauro et al. [80] devised the model using (w, u_{t-1}, u_t), where w is the total number of user requests observed per time period; u_t and u_{t-1} are the number of VMs allocated to the application in the current time step, and the previous time step, respectively;

Dutreilh et al. [81] followed the definition of the state as (w,u,p), where w is the total number of requests and u is considered as the number of VMs allocated and p as performance in the contract of average response time. Horizontal scaling has been done on the basis of three decisions: append the new VM, deallocate the VM and do nothing.

On the other hand, the resources assigned to each VM (CPU, RAM) for vertical scaling are considered in state definition. Authors Rao et al. [82] and Xu et al. [83] devised the state definition as $(mem_1, time_1, vcpu_1, ..., mem_u, time_u, vcpu_u)$, where mem_i, $time_i$ and $vcpu_i$ are the i^{th} VM's memory size, scheduler credit and number of virtual CPUs respectively. The possible operations for every three

variables can be increased, decrease or no-operation. The RIL defines the task as a combination of each variable operation. Rao et al. again proposed a technique, where RIL agent learned per VM [84]. State definition is configured as CPU, bandwidth, memory, and swap.

RIL learning is also very useful for tasks that are very closely related to auto-scaling problems, for example, the configuration of application parameters [85, 83]. Xu et al. [83] include the RIL agent with ANN to configure the parameters as per the application and VM performance, such as Session timeout, MinSpare-Servers, Keepalive timeout, Maxclient.

RIL can also be combined with control theory. Zhu and Agrawal [61] combine the RIL agent with a PID controller. Application parameters are guided by the controller to meet the Service Level Objectives (SLO). The resources are dynamically provisioned according to the parameters.

RIL algorithms are important to gain the best management policy for cloud applications without any initial knowledge. It is an online approach to learn and adapt as per the workload, application or system change. RIL technique could be a better approach to handle the auto-scaling for general applications, but the RIL approach is not mature enough to deal with the real cloud environment. This is an open research field and efforts are required to handle the flash workload and rapid change in state and actions.

- **Hidden Markov Model (HMM)** : This is modeled with a hidden Markov chain with the hidden states. In the hidden Markov chain technique, the state is known to the observers whereas in HMM it is invisible to observers. For example, two friends Sam and Rick are communicating on the phone. Sam describes his activities (eating, drinking) and Rick estimated the weather condition from his activities. The estimation is performed on the basis of the maximum likelihood estimation technique is shown in Figure 2.10. The parameters of HMM are x defines the states in the model, y is the total observations under consideration and state transition probability defined with a and b the output probability.

  An HMM model usually required fewer observations as compare to basic Markov chain models [86]. The system condition needs to observed minutely and input

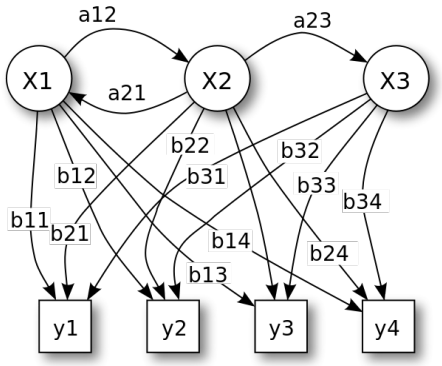

Figure 2.10: Hidden Markov Model (HMM) [6].

to the model. The author used the Weka data mining tool for implementing the HMM model, 2 states used with 0.01 iteration cutoff and spherical covariance. As per an author, model given 97% accuracy in the auto-scaling decision. Database tier is not considered in this work.

- **Support Vector Machine (SVM)**: It is also used in some articles. It is one of the supervised learning technique that is a combination of two techniques: classification and regression. It can be applied to non-linear workload. The classification is done on the basis of clusters formed and generates the classes for data under specific region. Linear SVM can be used for classifying the web workload.

  Liu et al. [87] proposed an adaptive technique with support vector machine (SVM) with linear regression (LR). Characterization has done on the basis of priority of the job and workload pattern. The generic model has been developed for cloud applications. The model can be extended further considering the different QoS parameters according to the SLA and application.

- Q-learning is the most used algorithm in auto-scaling by extending it with various techniques. Tesauro et al. [80] used the SARSA [79] approach. Some articles [82, 83, 84, 85] did not specify which machine learning approach used in their research, but problem definition and update function resemble the SARSA. There are many problems in Q-learning and SARSA addressed various ways as following [88, 89, 81, 83, 80]:

  – Initial performance is bad and required a large training period. On-line per-

formance is poor before the best solution is found. It requires the exploration of different states and actions.

- The curse of dimensionality problem in Q-learning. The number of state variables grows the state exponentially. States are stored in lookup tables. As the table grows, it takes time to search for the possible state from the lookup table.
- The environmental condition has a great impact on the performance of the RIL algorithms. As the conditions change the best optimal solution degrades the performance. RIL need to design to work with application behavior or environmental conditions.

Table 2.4 shows the taxonomy of reviewed articles in this section.

Table 2.4: Taxonomy on machine learning based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[80]	RIL+ ANN-SARSA + QM	P	H	Arrival rate, previous no. of VMs	Response time	on-demand	-	Trade application. Custom testbed.	Resource based
[82]	RIL+ANN	P	V	CPU and memory usage	Throughput, response time	on-demand	Custom tool	TPC-C, TPC-W and SpecWeb. Custom testbed. Xen.	VM performance based
[89]	TR + RIL	P	H	Request rate, no. of VMs, response time	Response time	on-demand	Custom tool. 20 seconds.	-	Resource aware
[61]	CT- PID controller + RIL + ARMAX model + SVR	P	V	Application adaptive parameters CPU and memory	App -lication related benefit function	on-demand	-	-	Resource and QoS aware

Continued on next page

Table 2.4 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[84]	RIL + CMAC	P	V	CPU, I/O, memory, swap	Response time	on-demand	Custom tool (Script)	TPC-C, TPC-W and SpecWeb applications. Custom testbed. Xen	SLA based/ Capacity based
[81]	RIL	P	H	No. of user requests, no. of VMs, response time	Response time, cost	on-demand	-	Olio application. Custom testbed.	resource-aware
[83]	RIL + ANN	P	V	CPU and memory usage	Response time	on-demand	Custom tool	TPC-C, TPC-W, SpecWeb applications. Custom testbed.	Budget aware/QoS based
[88]	RIL	P	H	No. of user requests, no. of VMs, response time	Response time, cost	on-demand	Simulated	Custom simulator (Matlab)	Workload aware
[85]	RIL + Simplex	P	V	CPU, memory - application parameters	Response time, throughput	on-demand	Custom tool	-	Resource aware
[86]	HMM	R	H	No. of request, MAPE, RMSE	Cost-Performance	on-demand	-	-	SLA aware

Continued on next page

Table 2.4 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[87]	LR (Linear regression) + SVM	R	H	Prediction time (ms)	Cost-Performance	on-demand	Custom tool for short, medium and large task length	Simulation	Workload based

2.10.5 Queuing Theory

It is one of the widely used for modeling Internet applications. It is used in the analysis phase of the auto-scaling process. It estimates the performance metrics and waiting time for the requests. Queuing theory is a field of applied probability to solve the queuing problem. Queuing issue is quite common in many fields such as telephone exchange, petrol station, supermarket etc. The structure of the model is shown in Figure 2.11. The requests arrive at the server at mean arrival rate λ and remain in queue till the processing. One or more servers are available to process the requests at the mean process rate μ.

Figure 2.11: A simple queuing model with one server [7].

Author Kendall [90] represents the standard notation for queuing model known as kendall's notation. It describe the queue as $\mathscr{A}/\mathscr{B}/\mathscr{C}/\mathscr{K}/\mathscr{N}/\mathscr{D}$.

- $\mathscr{A}$: Arrival process.
- $\mathscr{B}$: Service time distribution.
- $\mathscr{C}$: Number of servers.
- $\mathscr{K}$: Capacity of system. The number of places in the system.

- $\mathscr{N}$: Calling population. Size of users from where the request comes. Open population has an infinite number of customers and closed model is a finite number of customers.

- $\mathscr{D}$: Queue's discipline. It represents the priority of jobs.

$\mathscr{K}, \mathscr{N}, \mathscr{D}$ elements are optional, by default variables are considered as $\mathscr{K} = \infty$, $\mathscr{N} = \infty$ and $\mathscr{D} = FIFO$. FIFO (First In First Out) is mostly used, which served the request as they come. Another important one is Process Sharing (PS). Markovian (M) refers to a Poisson process characterized by λ, which indicates the number of arrival request per time unit. D stands for deterministic also refers to constant. G is also used commonly known as general distribution.

Multi-tier applications are complex in nature and the queuing network can be useful. The load balancer is represented by a single queue and distribute the coming requests to the VMs.

Queuing theory is useful for stationary nature systems. It can work with both proactive and reactive kind of environment. The main objective of the cloud-based system is to develop a model on the basis of some known parameters (e.g. Arrival rate λ). Performance metrics measured as the mean response time, and the average waiting time in the queue. The web application workload is dynamic, it requires the timely recalculation of queuing model and metrics.

The queuing model can be used in two ways: simulation and analytical method. The analytical approach can be used with simple models. $M/M/1$ and $G/G/1$ are the well-known methods used to define the arrival and service process. Simulation can be used for the complex system to obtain the desired metrics.

$M/M/1$ (Poisson-based) is a basic queuing model. Exponential distribution is followed by both the arrival times and the service times. The mean response time R of $M/M/1$ model can be calculated as $R = \dfrac{1}{\mu - \lambda}$. Arrival rate is represented by λ and μ shows the service time respectively. $G/G/1$ is another simple method. Inter arrival time and service time are controlled by general distributions with prior information of mean and variance. $G/G/1$ system is represented by the following equation:

$$\lambda \geq \left[s + \frac{\sigma_a^2 + \sigma_b^2}{2(R-s)} \right]^{-1} \tag{2.4}$$

R is the mean response time and s is average service time. The variance of inter-arrival time is σ_a^2 , σ_b^2.

Little's Law [91] is also used in many queuing scenarios. It states that the average number of requests $E[C]$ in the system is same as the average customer arrival rate λ multiplied by the average time of each customer in the system $E[T] : E[C] = \lambda \times E[T]$.

Simple and complex queuing used in the research articles for analysis of the performance of applications and system.

Ali-Eldin et al. [63, 62] used a $G/G/n$ queue to model a cloud application, and n represents the number of servers. The model used to calculate the required resources to process the hosted application workload λ, the response time according to the configuration of servers.

An elastic cloud application can be represented using the queuing network, considered each VM as a separate queue. Urgaonkar et al. [92] devised a technique using $G/G/1$ queues. Histogram used to predict the peak workload. The number of servers calculated as per the queuing model and peak workload in specific time step. The reactive approach further used to correct the value. The deficiency of the technique is the under-utilization of resources.

Multi-tier applications can be deployed in a cloud environment and assigned one or more queues for a specific tier. Zhang et al. [93] proposed a closed system with a network of queues, which handles a limited number of users. Han et al. [94] deploy the multi-tier application as an open system. $G/G/n$ queues have been used and considered one queue per each tier. Bacigalupo et al. [95] used the queuing model for the three-tier web application. It computes the response time per each tier.

The models discussed are considered the analysis part of the MAPE-K loop. Some techniques are used to implement the planning phase using optimization algorithm and predictive controller [62] in order to maximize the revenue for the datacenters [96]. On-line monitoring captures the number of requests, which further input to the queuing model to estimate the number of VMs required for the processing of application workload [94, 92]. Zhang et al. [93] proposed a model using regression technique to approximate estimate the number of CPU by calculating the quantity of a client requests at each tier (e.g. Browsing, ordering or shopping).

Application of the queuing model is for cloud application and system modeling. It

defines the static architecture and required an update in parameters or structure of the model. Simulation and the analytical tool can be used to solve the model. Any change in the number of input request or a number of resources needs to update the model, which is not cost-efficient. Most of the articles used CloudSim simulator [97], Vondra and Sediv [98] developed a new simulation tool specifically for the auto-scaling. It includes the trace file of widely used workloads.

Queuing model is a component in auto-scaler. The model works quite efficiently with linear parameterized applications, and not fit for the applications with a non-linear workload. The queuing model requires more efforts to work with multi-tier application because of complex nature and non-linear relationships. Table 2.5 shows the taxonomy of reviewed articles in this section.

Table 2.5: Taxonomy on queuing theory based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[96]	QTH	R	H	Arrival rate, service time	Response time	on-demand	Simulated	Custom simulator (Monte-Carlo)	Budget Aware
[93]	QTH + Regression (Predict CPU load)	P	-	CPU load, Number and types of transactions	-	on-demand	Custom tool. 1 minute	Custom simulator, based on C++Sim. + Data collected from TPC-W	QoS Based
[92]	QTH + Histogram + Thresholds	R/P	H	Peak workload	Response time	on-demand	Custom tool. 15 minutes	Custom testbed. Xen + 2 applications (RUBiS and RUBBOS)	Resource Aware
[95]	QTH + Historical performance model	P	H	Arrival rate	Response time	on-demand	-	Custom testbed. Eucalyptus + IBM Performance Benchmark Sample Trade	Prediction /Resource Aware

Continued on next page

Table 2.5 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[94]	QTH (model) + Reactive scaling	R	H	Arrival rate, service time	Response time, cost	on-demand	Custom tool. 1 minute	Custom testbed (called IC Cloud) + TPC-W benchmark	Cost-Aware
[99]	QTH + kalman Filter	P	H	Arrival rate	Response time	on-demand	Custom tool. 10 seconds	Custom TestBed	User Aware
[100]	QTH	P	H	Arrival rate	Response time, Bottle-necks	on-demand	-	Custom testbed. Open stack.	QoS based
[98]	QTH (Sim-ulator for auto-scaling)	P	H	Arrival rate	VM uti-lization, Arrival rate	on-demand	Custom tool. 15 minutes	Custom Simu-lator for gen-eral purpose.	-

2.10.6 Threshold-based Rules

Threshold based techniques are simple to implement and very popular. The threshold values required a deep understanding of all the parameters of the corresponding environment and workload. Amazon EC2 [44] and RightScale [45] are using threshold-based technique. The threshold based rule technique is purely related to planning. The number of resources allocated to the application in the form of VM according to a set of rules. There are two types of scaling decision: Scaling up/down vertically increase or decrease the capacity of the working node. Scaling in/out refers to horizontally increase or decrease the VM capacity. The scaling up/out and scaling down/in is working on the principle of Algorithm 1.

There are two parts of each formula: condition and action. $x1$ and $x2$ are performance metrics (e.g. Number of requests, response time, budget, CPU load etc.), tU and tL are the upper and lower threshold values of the performance metrics. Threshold conditions checked with upper duration dU and lower duration dL, which considered in the seconds. If certain conditions are met, the action will be taken. The manager

Algorithm 1 The algorithm of threshold rule based auto-scaling

```
1: if x1 > tU1 and/or x2 > tU2 and/or ... then          ▷ tU_i is upper threshold
2:     if Clock % dU == 0 then                          ▷ dU is upper duration
3:         n = n + r                                    ▷ Scale-out or Scale-up
4:     else if Clock % iU == 0 then                     ▷ iU is upper cool down period
5:         do − nothing
6:     end if
7: end if
8: if x1 < tL1 and/or x2 < tL2 and/or ... then          ▷ tL_i is upper threshold
9:     if Clock % dL == 0 then                          ▷ dL is lower duration
10:        n = n − r                                    ▷ Scale-in or Scale-down
11:    else if Clock % iL == 0 then                     ▷ iL is lower cool down period
12:        do − nothing
13:    end if
14: end if
```

should decide resources r acquired or released during the action performed from the total resources n. During horizontal scaling r is the number of VMs and during vertical scaling r is CPU or RAM. The lower and upper cool down period is set as iL and iU, during this time auto-scaler do nothing.

It is easy to deploy a threshold-based technique in a cloud environment. Defining rules are a difficult task, it requires input from the client on different QoS metrics. QoS parameters have performance trade-off. Application manager has to give threshold value of performance metrics parameters. The experiment carried out by Koperek and Funika [101], they defined the performance benefits of the application-oriented metrics than system specific metrics. The threshold needs to be carefully decided otherwise it can cause the oscillation problem [89]. To handle the oscillation issue, certain techniques such as cool down, calm and inertia periods are set, so no scaling decision can take place in these time slots.

Most of the techniques are using one or maximum two performance metrics. Commonly used performance metrics are input request rate, CPU load time and average response time of application. Some authors are taking application response time as a performance metric [89, 102] while few focused on the system level metrics such as

storage, network, and CPU [103]. Author Mahallat [104] considered the scaling overhead and SLA violation. Due to the introduction of a new type of resources author Qu et al. [53] evaluate the availability metrics along with response time.

The number of threshold values also varies with different techniques. Most of the providers are using two threshold values: Upper and the lower threshold value. Hasan et al. [103] used four threshold values. In this technique along with upper threshold $(ThrU)$, the author defines $ThrbU$, which is marginally lower than $ThrU$ and $ThroL$ is pretty above than the lower threshold $(ThrL)$. The significance of setting such a threshold is to determine the trends. Scaling action could be taken as per the trends.

Metrics are fetched from the monitoring tool. Collecting the run-time data from the monitor and taking action as per the rules is a reactive approach.

RightScale [45] voting system is combined with the reactive approach. If most of VMs agree to scale up/out or down/in decision, then scaling action will be performed. Threshold needs to set by the application manager. Several author are using the RightScale technique (e.g. Simmons et al. [105], Kupferman et al. [30], Ghanbari et al. [25], Chieu et al. [106]). Author Chieu et al. [106] working on the scalability of web applications by defining the set of rules for the active sessions, further extend his work [107] using RightScale. The rules are decided as per the upper and lower threshold value of the sessions. If the sessions are going upper or lower scaling decision will be taken.

RightScale works on the voting system, but it is highly dependent on the threshold values set by the application-manager and the nature of the application workload. Kupferman et al. [30] compared the RightScale with another technique and concluded the disadvantages of RightScale. Simmons et al. [105] attempted to overcome this issue using the strategy tree. The regression-based model used for three types of scaling policies. Strategy tree follows the parent as per the workload trend.

The threshold rule-based technique is popular due to its simplicity and the client can easily understand. Reactive nature of TR technique is a major challenge. Another issue in the TR technique is to set appropriate performance metrics. In a cloud environment approximate startup time of the VM is 5 to 10 minutes, so to ready the virtual machine in a reactive manner put a delay in service, which leads to performance degradation. To resolve this issue to some extent, Lorido-Botrcn et al. [108] suggested the dynamic

threshold values. The initial values of the threshold are static and later dynamic threshold values set as per the SLAs violation.

Multi-cloud scaling is another issue in the auto-scaling. Auto-scaling for three-tier application has been developed by Grozev and Buyya [109]. The rule-based technique has been devised to take scaling decision for multiple data center.

Along with the on-demand resources, spot-instances are used for web application provisioning. The spot instances are 90% cheaper than on-demand resources. So effective scaling strategy can provide significant benefit to the clients as well as a service provider to reduce the SLA violation. Qu et al. [53] provided a reliable solution to use spot instances for web application provisioning. Spot-instances have a potential for fault tolerance and tackling of web application's flash crowds or flash events workload.

Smart kill [110] is an important idea used to reduce the cost. Most of the service provider charge hourly basis. It can further improve the system performance to avoid VM starting and shutdown time. It can reduce the SLA violation.

It is easy to implement the rule-based auto-scaling of the particular application. If the workload and nature of application can forecast easily than rule-based technique can be used. Application with an unpredictable pattern should choose other suitable scaling strategies. Table 2.6 shows the taxonomy of reviewed articles in this section.

Table 2.6: Taxonomy on threshold rules based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[30]	RightScale and TSA (LR + AR(1))	Mix	H	Request rate, CPU load	-	on-demand	Simulated	Custom simulator with real experiment	-
[105]	RightScale and Strategy tree	R	H	CPU idle time, no. of sessions	-	on-demand	Custom tool(4 minutes)	Real provider. Amazon EC2 + RightScale + web app.	Policy Based
[101]	TR	R	H	Average waiting time in queue	CPU load	on-demand	Custom tool	Public cloud: FutureGrid, Eucalyptus India cluster	QoS based

Continued on next page

Table 2.6 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[107]	RightScale + MA to performance metric	R	H	Number of active sessions	-	on-demand	Custom tool	Custom testbed. Xen + custom collaborative web application	Perfor-mance Based
[25]	RightScale	R	H	CPU load	-	on-demand	Amazon Cloud-Watch	Real provider. Amazon EC2 + RightScale (PaaS) + web application	Qos / Perform-ance based
[111]	TR	R	V	CPU load, memory, band-width, storage	-	on-demand	Simulated.	Custom simulator, plus Java rule engine Drools	SLA based
[102]	TR	R	Both	CPU, memory, I/O	Response time	on-demand	Custom tool. 1 minutes	Custom testbed (called IC Cloud) + TPC	Cost Aware
[103]	TR	R	Both	CPU load, response time, network link, jitter and delay	-	on-demand	-	-	Storage/ Network aware
[110]	TR + QTH	P	H	Request rate	Response time	on-demand	Amazon Cloud-Watch. 1 – 5 minutes	Real provider. Amazon EC2 + Httperf + MediaWiki	QoS Aware
[108]	TR	R	V	CPU load	Response time	on-demand	Simulated. 1 minute	Custom simulator	Cost and SLA Aware

Continued on next page

Table 2.6 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[109]	TR	R	H	CPU load	Response time	on-demand	Custom tool and manual.	Amazon EC2 (4 data centers)	Cost aware
[104]	TR + Learning Automata	R	H	Scaling Overhead	SLA violation	on-demand	Simulated. 5 minutes	CloudSim	SLA aware

2.10.7 Time Series Analysis

Time series analysis is a very popular model and has applications in many areas including economics, bioinformatics, finance, engineering, etc., to predict the measurement for future time steps. An example, the number of requests that reach the server for an application at one-minute intervals. The time series is used in the analysis phase to forecast future workload.

Performance metrics (input workload, CPU load) sampled at fixed time intervals (e.g. 5 minutes). The predicted series X gained from the sequence of observation w.

$$X = x_t, x_{t-1}, x_{t-2}, ..., x_{t-w+1} \tag{2.5}$$

Time series forecasting applied in auto-scaling to estimate the future workload or resources. Predefined rules are designed in planning phase [112], and optimize the resource allocation [68].

As discussed earlier, the main objective of time series analysis is to predict the future workload, on q observation from historical workload known as a history window or input workload (where $q \leq w$). Time series analysis classified in two categories: direct prediction and identification of a pattern in time series. The first category, direct prediction contains auto-regression, moving average, ARMA, ARIMA, exponential smoothing, and machine learning approaches:

- Moving Average (MA): A widely used model to smooth a time series to filter noise from random fluctuation and to make predictions. General formula of MA is as follows:

$$\hat{x}_{t+r} = a_1 x_t + a_2 x_{t-1} + ... + a_q x_{t-q+1} \tag{2.6}$$

$\hat{x}_{t+r}$ is a forecast value from last q observations with weighted average. Prediction interval denotes by r, and typically sets to 1. Equal or different weight factors are assigned to the values denoted by $a_1, a_2, a_3, ..., a_q$. There are different types of MA's are Simple Moving Average (SMA) and the Weighted Moving Average (WMA). Moving average performs in various forms, but the objective of MA is the same. In simple moving average's general formula is considering the arithmetic mean of previous q values. It considers the same weight of all the observations. WMA considers the different weight for each observation. The highest weight assigned to new observation, while less weight is given to previous observations.

- Exponential Smoothing (ES): In contrast to the moving average, the weighted average of previous observation is also calculated, but the ES considers all the observation of time series (w values) from past history. The new parameter α, the smoothing factor has very less influence on the predicted value, because exponentially decreasing weight assigned to the new observations. There are many versions of ES such as simple ES and Brown's double ES [113]. The smoothed value s_t is calculated from the present observation and past smoothed value based on recursive formula: $s_t = \alpha x_t + (1-\alpha)s_{t-1}$. Time series with fewer trends can be forecast using simple ES. Brown's double ES is suitable for linear trend time series. Two smoothing series are required to calculate as:

$$\begin{aligned} s_t^1 &= \alpha x_t + (1-\alpha)s_{t-1}^1 \\ s_t^2 &= \alpha s_t^1 + (1-\alpha)s_{t-1}^2 \end{aligned} \tag{2.7}$$

s_t^2 is calculated from simple ES to s_t^1. The trend d_t and level c_t is obtained from s_t^1 and s_t^2 smoothed values. The forecast value $\hat{x}_{t+r}$ is obtained from following formula:

$$\begin{aligned} \hat{x}_{t+r} &= c_t + r_{dt} \\ c_t &= 2s_t^1 - s_t^1 \\ d_t &= \frac{\alpha}{1-\alpha}s_t^1 - s_t^2 \end{aligned} \tag{2.8}$$

- Auto-regression AR(p): The future value is forecast using the linear weighted

sum of previous observation p in the time series:

$$x_{t+1} = b_1 x_t + b_2 x_{t-1} + ... + b_p x_{t-p+1} + \varepsilon_t \tag{2.9}$$

p represent the number of observations in the AR equation, which could be different from history window w length. White noise ε_t is added in the formula. AR coefficients are calculated using different methods such as maximum likelihood or least squares. The widely used technique in the literature is Yule-Walker equations and auto-correlation coefficient. The formula of auto-correlations as follows for x_t to x_{t-k}, where $k = 1, 2, 3...$:

$$r_k = \frac{covariance(x_t, x_{t-k})}{var(x_t)} = \frac{E[(x_t - \mu)(x_{t-k} - \mu)]}{var(x_t)} \tag{2.10}$$

- Auto-Regressive Moving Average, ARMA(p,q): This model is the hybridization of both AR of order p and MA of order q. ARMA model is described as:

$$x_t = b_1 x_{t-1} + ... + b_p x_{t-p} + \varepsilon_t + a_1 \varepsilon_{t-1} + ... + a_q \varepsilon_{t-q} \tag{2.11}$$

ARMA model can be used either purely as AR or MA model using ARMA$(p, 0)$ and ARMA$(0, q)$ respectively. ARMA is best suitable for stationary time series.

The extension of the ARMA model is ARIMA (Auto-Regressive Integrated Moving Average) model is suitable for non-stationary time series. If ε (white noise) shows no pattern, then ARIMA(p, d, q) used where d represents the degree of difference.

- Machine Learning Techniques: Analysis of time series was carried out by many authors using machine learning-based techniques.

Regression-based techniques are based on statistical methods to form a polynomial function, that find the nearest points from the history window w. Linear regression is ordered 1 polynomial expression. The distance of points should be as less as possible. Multiple Linear Regression is used when there is more than one variable in the expression.

Neural Network is a group of interconnected artificial neurons put on multiple layers. Multiple inputs are put in the hidden layer, which further given an output.

In time series one output against one input from the history window. Random weights are assigned to the input vector during the training phase. The weights are adapted to the optimized output has not achieved.

- Pattern Recognition: Time series data is a combination of seasonal and non-seasonal data. The time series has patterned with a specific time period (e.g. hours, day, month, year, or season) on the basis of short term and long term workload. It finds the matching pattern in the history that relates to the current pattern. It is very similar to the string matching technique [114].

- Signal Processing Techniques: This technique is based on harmonic analysis to decompose a time series signal into different frequencies. Fast Fourier Transformation and spectral density estimation filter the noise and estimating the signal value.

- Auto-correlation: In the linear regression errors are independent, the auto correlation function is used to deal with the dependent error pattern. The input workload from the history window is shifted recursively, and further compared with the original time series.

The histogram is used to represent the time series. It splits the time series values into equal size bins, and each bin represents the frequency. In literature, it is used to represent the forecast values, resource distribution, and usage pattern.

Review of Articles

In the literature, time series analysis is mostly used prediction model for multi-tier applications. A simple moving average model gives poor results [115], so the MA used to remove the time series noise [59, 60]. Huang et al. [116] devised a resource prediction model using double exponential smoothing, and simple mean and Weighted Moving Average (WMA) applied for comparison. ES significantly gives better results because of history records w. Mi et al. [117] applied Brown's double ES to predict the workload and achieve a good result for HTTP workload with a small error.

The auto-regression technique has also applied for workload and resource prediction [118, 119, 115, 30, 68]. Roy et al. [68] used the AR model for workload forecasting

by taking the previous three observations. Further response time estimated from the predicted values. An optimization controller applied to find resource allocation, considering SLA violation cost, reconfiguration, and leasing resources. Kupferman et al. [30] used AR(order 1) to forecast the requests per second, and concluded that its performance highly lies on many manager-defined parameters (e.g. Size of history window, size of adaption window, monitoring-interval length). The forecasting is determined for short-term and long-term trends, it is highly depended on the size of the history window. Adaptation window determines the future model extension.

ARMA model is a simple and efficient model to predict the future workload (number of requests). Fang et al. [120] predict VMs CPU usage. ARIMA model is applied in various articles [121, 122, 123]. ARIMA required historical workload. The performance of the model highly depends upon the history window. ARIMA approach is ideal for dynamic workload such as web applications. Sedaghat et al. [121] applied the horizontal and vertical scaling to increase the benefit in terms of cost. Mao and Humphrey [124] used the classification given as: increasing, stable, seasonal and on/off. Repacking (or reconfiguration) of VMs to provide certain capacity, and repacking of the application based on workload has been done. The approach then finds the optimal pack of VMs and applications. The proposed approach is able to save 7% to 60% cost for resource utilization. The container-based approach can further be optimized by considering more QoS parameters. Calheiros et al. [122] used ARIMA model for workload prediction, and evaluate the impact on different QoS parameters. The web application workload is dynamic and contains seasonal data. The model gives 91% accuracy for non-seasonal data, but not fit for the highly non-seasonal workload. This work can be further extend using an adaptive approach for classification of workload, and design the heuristic for ARIMA fit function for different classes. As discussed earlier, one model doesn't fit for all types of workload, Messias et al. [123] present GA based approach for time series prediction. Traces of real workload have been used to evaluate the prediction model. A new metric has been introduced in the article named as an Elasticity Index (EI), which describe the solution optimization. The range of EI varies from (0 to 1), a value near to 1 means the solution is good. The model gives less error as compared to other models. This approach is taking more time to predict the incoming request even prediction is done on an hourly basis. Further, this model can be extended by mapping a

few prediction techniques with a specific application and workload pattern. GA model can also design, particularly for cost, energy, sharing of resources, parameters.

The accuracy of neural network [125, 126] and multiple regression equation [65, 126, 30] model are highly dependent upon on the size of input the history window. Islam et al. [125] used more than one value from the history and got a better result. Kupferman et al. [30] devised the necessity of balanced size of input history window. Regression of various window sizes applied to find the prediction values. The prediction interval of r is also an important factor. Islam et al. [125] investigated the size of the interval window and found appropriate scaling interval is of 12 minutes, because of VM startup time is between 5-15 minutes. Prodan and Nae [126] applied the neural network to forecast the game load for 2 minutes. In contrast, the neural network is better than MA and ES in terms of accuracy.

Horizontal and vertical scaling is also an important factor considers by many authors in literature. It can be either taken separately or in a hybrid manner also. Dutta et al. [127] investigated that horizontal scaling has higher configuration cost as compare to vertical scaling, but relatively have larger throughput. The author prefers the horizontal scaling. Regression model applied to estimate the future workload. Fang et al. [120] focused on horizontal scaling (CPU and memory) to handle the flash crowd, whereas vertical scaling applied for regular changes.

Proactive time series forecasting can be combined with a reactive approach. Iqbal et al. [5] devised a hybrid model for auto-scaling, the author uses the reactive technique for scale-up and a regression model for scale-down. Polynomial regression used to calculate the number of application-tier and database-tier VM instances.

A pattern identification technique also applied on time series analysis by some authors [128, 115, 129]. Gong et al. [115] proposed an FFT based technique to identify the matching patterns in resource utilization (CPU, RAM, Network and I/O). Auto-correlation, histogram, and auto-regression are used for the comparison. Caron et al. [128] designed the algorithm with more number of parameters, which gives poor performance.

Resource utilization of application is also estimated with the simple histogram in some articles by using mean distribution [118], and the highest frequency in the bin [115]. The dynamic load balancer is introduced using Holt's approach by using current

and historical data [130]. This work can be used with web workload, which may give prediction efficiency and helps to resolve the load balancing issues.

Time series analysis techniques are able to forecast the future workload of web applications. Further, this information can be used to predict resource requirements. This technique is very appealing because of input workload is known to the auto-scaler in advance, and have enough time to prepare the VMs beforehand. The drawback of the technique is the accuracy, which depends upon the input workload, history window selection, metrics, prediction interval, and target application. There is no best solution for all types of time series forecasting. The future scope of time series forecasting for adaptive or GA based technique for time series forecasting for a specific application by considering the QoS parameters. Table 2.7 shows the taxonomy of reviewed articles in this section.

Table 2.7: Taxonomy on time series analysis based reviewed literature

Ref.	**Technique**	**R/P**	**H/V**	**Metric**	**SLA**	**Pricing Model**	**Monitor**	**Experiment**	**Approach**
[118]	TSA: AR(1) and Histogram + QTH	P	H	Request rate and service demand	Response time	on-demand	Simulated. 1, 5, 10 and 20 minutes	Custom simulator + algorithms in Matlab	Resource Aware
[119]	TSA: AR	P	H	Login rate, number of active connections, CPU load	Energy consumption, Service not available	on-demand	Simulated	Simulator.	Energy Aware
[126]	TSA: ML Neural Network	P	H	Number of entities (players)	Prediction accuracy	on-demand	2 minutes	Simulator of a MMORPG game	QoS based
[117]	TSA: Brown's DES.	P	-	No. of requests per VM	-	on-demand	10 minutes	Custom testbed. TPC-W	Performance based

Continued on next page

Table 2.7 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[115]	TSA: FFT and Discrete Markov Chains	P	V	CPU load	Response time	on-demand	Libxenstat library. 1 minute	Custom testbed. Xen + RUBiS + part of Google Cluster Data trace for CPU usage.	SLO based
[112]	TSA: AR + TR	P	Both	Number of requests	-	on-demand	Zabbix	Xen + Eucalyptus + PhpCollab application. Custom testbed + Amazon EC2.	Resource Aware
[128]	TSA: Pattern matching	P	H	Total number of CPUs	Number of serviced requests, cost	on-demand	100 seconds	Analytical models	Resource Aware
[129]	TSA: FFT and Discrete-time Markov Chain	Mix	V	CPU load, memory usage	Response time, job progress	on-demand	Libxenstat library. 1 second	-	SLO Aware
[68]	TSA: AR	P	H	Number of users in the system	Response time, VM cost, application reconfiguration cost	on-demand	-	No experimentation on systems	QOS Aware

Continued on next page

Table 2.7 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[5]	TR + TSA, polynomial regression	Mix	H	CPU load, number of requests, number of VMs	Response time	on-demand	1 minute	-	SLA based
[120]	TSA: ARMA	P	Both	Number of requests, CPU load	Prediction accuracy	on-demand	-	Custom testbed. Xen and KVM	SLA based
[116]	TSA: Brown's double ES. Compared with WMA	P		CPU load, memory usage	Prediction accuracy	on-demand	Simulated	Custom testbed. TPC-W	Cost-Aware
[125]	TSA: ML Neural Network and LR + Sliding window	P	H	CPU load (aggregated value for all VMs)	Prediction accuracy	on-demand	Amazon CloudWatch. 1 minute	Real provider. Amazon EC2 and TPC-W application. Experiments in R-Project.	Resource Aware
[127]	TSA: Polynomial regression	P	Both	Number of requests	Response time, cost for VM, application license and reconfiguration	on-demand	Custom tool. 1 minute	Custom testbed. KVM + Olio	Resource / Cost Aware

Continued on next page

Table 2.7 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[121]	TSA: ARIMA + Repacking	Mix	Both	Number of request, cost	Response time	on-demand	4 and 20 minutes.	Custom simulator	Cost-Aware
[51]	TSA + Profiler	Mix	H	Number of request, CPU usage and throughput	Response time	on-demand (Heterogeneous)	5 minutes	-	QoS Aware
[122]	TSA: ARIMA	P	H	Request rate, RMSE	Response time	on-demand	simulated	CloudSim toolkit	QoS aware
[123]	TSA: ARIMA + GA	P	H	Request rate, Bootstrap, Mean Elasticity Index (MEI), RMSE, MSE	Response time, Cost	on-demand	simulated	Custom testbed in cloud	Resource Aware
[130]	TSA: Holt model + Reverse trend (RT) + GA + Fuzzy logic	P	H	Average and standard error and Breakdown	Execution time, Number of migrations	on-demand	controlled environment	Custom testbed using two clusters with Globus toolkit	Load Balance aware

Continued on next page

Table 2.7 – *Continued from previous page*

Ref.	Technique	R/P	H/V	Metric	SLA	Pricing Model	Monitor	Experiment	Approach
[131]	TSA: Double Exponential Smoothing (DES)	P	H	Number of requests	Resources and SLA	on-demand	Custom	CloudSim	Cost aware
[132]	TSA: Weight moving average (WMA)	Mix	H	Number of requests + CPU utilization	cost	on-demand	Custom	CloudSim	cost aware

2.11 Challenges in Multi-tier Applications Resource Management

Challenges come with opportunities. Effective resource management for multi-tier application is a challenge in the cloud environment.

- The different classes of resources have a different effect on the QoS. CPU share has a larger effect on the QoS than different resources for processing intensive applications of computing. In other cases, the larger impact can be on memory and disk. In any case, the relation between the QoS and resources is exceptionally non-linear and challenge to determine execution[133, 134].

- The resource provisioning in multi-tier applications is a more complex task as compared to the single-tier application because of the way that the resource request at every level is distinctive. The interaction between different tiers may lead to an impact on QoS. So provisioning becomes a difficult task as it is hard to decide when to provision and in what amount.

- The coarse-grained and wrong methodologies might cause low resource utilization or performance degradation and leads to a penalty. Many existing technique use VM migration but it is not fit for all types of application [118, 135, 136].

- The performance guarantee is not provided by cloud providers in regard to application level performance. Additionally, the bandwidth of I/O including network and disk is unpredictable and dynamic[137, 138]. Different economic or pricing methods can have an enormous effect on multi-tier resource management. The relationship between economic and distributed systems in cloud computing is analyzed [137, 138]. SVM is used for the prediction of the interference score for various I/O workloads all together to offer better reasonableness for client properties of recorded estimations.

  Many existing methods can enhance the efficiency of resource management. But still, the application performance is not controllable. For example, Xiong et al. [135] studied the relationship between throughput and response time by the number of simulations. However, results depict that mean response time is different for the same set of resources. Therefore, there are many hidden metrics that affect the performance of the system.

2.12 Summary

Auto-scaling technique reserves the resources as per the incoming workloads. The efficiency of the algorithm highly depends upon the prediction model. QoS parameters are considered while designing the auto-scaler, which leads to SLA fulfillment and avoid the penalty cost. Existing auto-scaler are still facing many research challenges. Research articles have identified the issues and define with different characteristics.

In this study, a state-of-the-art survey has been carried out on auto-scaling and resource provisioning techniques and identified the key challenges in the resource provisioning of the multi-tier web application. Auto-scaling techniques have been classified, and taxonomy describes the parametric similarities and differences in various auto-scaling techniques. As per the review of articles, suggest some research directions for the research community.

CHAPTER 3

A TECHNOCRAT ARIMA AND SVR MODEL FOR WORKLOAD PREDICTION

3.1 Introduction

Cloud computing becomes a potential platform for the services of the next generation. The cloud-based organizations like Google, Amazon, Microsoft, and IBM are providing services like Software-as-a-Service (SaaS), Infrastructure-as-a-service (IaaS), and Platform-as-a-Service (PaaS). As users are moving business applications on the cloud environment, it increases the competition among the Cloud Service Providers (CSP). The cloud resource provisioning (e.g. bandwidth, memory, type of VM, etc.) affects the Quality of Service (QoS) from different aspects such as reliability, response delay, etc. [139]. Thus, CSPs must give the guaranteed QoS to the clients to sustain in the cloud landscape.

A firm set of machines acquired in the static allocation of resources. The web application has a different incoming workload pattern in a cloud environment depending on the minute, hour, day, week, month and year. The static allocation mechanism faces resource oscillation problems due to different access patterns.

Cloud computing has the elasticity characteristic, which provides the dynamic resource supply on-demand. The resources can be acquired or released according to the requirement as per the incoming requests to the application. The CSPs like Amazon Web Services (AWS) and Google App Engine have the scalability characteristic that could enhance resource usage and improve the QoS of the end-users. The scalability feature increases or decreases the capacity of computing and storage.

The deployment of the requested number of resources is a challenge in scalability to provide QoS to the end-users. In the literature, the auto-scaling addressed two types of techniques. The first approach is reactive, which takes the scaling decision based on the threshold values for resource utilization in cloud infrastructure. The second approach is a proactive technique which predicts the future workload to scales resources prior to the incoming workload. The reactive approach for auto-scaling was deployed by most cloud providers. However, due to the high fluctuations in the web application workload, this system faces high resource oscillation issue. In the case of the flash event or flash crowd, the reactive approach raises the problem of *Slashdot effect* [140]. Therefore, a need arises for a methodology capable of capturing the various pattern of the workload with non-seasonal and seasonal traffic.

3.2 Related Work

In literature, there are two methods to solve the auto-scaling in the cloud landscape. First, the resource utilization in the data centers is measured. Afterward, the threshold rules are designed for scale-out or scale-in of resources configuration by the client or human expert. Another approach is the autonomic technique with workload forecaster. The performance modeler estimated the required resources for the future incoming workload. The cost and Service Level Agreement (SLA) trade-off required serious consideration while designing the provisioning technique.

The characteristic of web application traffic is analyzed in recent research articles. In this thesis, the researcher focused on proactive approaches to predicting web traffic in the form of time series through Machine Learning (ML) and statistical models. The workload classification applied to select the appropriate model according to the time series pattern present in the sliding window. This section describes a brief of the workload characterization and prediction approaches.

3.2.1 Workload Characteristics

The classification and characterization of the incoming requests pattern is a valuable tool for understanding the workload of cloud applications. In this field, there are a lot of research articles.

Panneerselvam et al. [141] classified the workload as once-in-a-lifetime, constantly changing, static, periodic and unpredictable into 5 different categories. Workload characteristics investigated from workload types such as compute-intensive or data-intensive jobs. The performance of Bayesian classifier on CPU-intensive jobs is satisfactory, and memory-intensive jobs are better predicted by the Markov model. The average prediction error of the models is 36% to 58%, which is quite high.

Zhang et al. [142] developed a web-based application workload factoring technique. It divided the incoming workload as flash-workload and base-workload. It classified the applications according to content and volume prominence. A threshold-based approach applied to factor the workload into the base and crowd zone of the flash workload. The model deficiency is that it can fit only in the hybrid cloud infrastructure.

Eldin et al. [143] used the statistics, time series analysis and polygon splines on the Wikipedia traces to analyze the seasonality and trend. The strong seasonality is present in the workload which is highly predictable. There is good efficiency in the short-term prediction, but experienced lower accuracy for the workload contains the spikes.

Patel et al. [144] developed a clustering-based technique to recognize resources usage within clusters for various patterns in the workload. The application trace log selected randomly and identified the workload pattern. The limitation of this model is that at the initial stage, it requires historical observations that make it difficult to estimate the resource at the beginning stage. The global optimization in the data center can improve resource management in the cloud environment.

Wang et al. [145] researched on the influence of workload characterization on slow time-scale non-stationarity and rapid time-scale stochastic. This model helps to analyze the workload traces to choose the dynamic resizing. The crucial factors for dynamic resizing such as reliability, storage, and energy are not considered in this article.

The synthetic load generator (BURSE) for cloud computing developed by Yin et al. [146]. The workload with burstiness and self-similarity can be produced with the BURSE. Perhaps, the actual workloads have highly relied on the application and change

with time on a regular basis. In this method, the dynamic characteristics of the workload have not considered.

Calzarossa et al. [147] analyzed the conventional web workload patterns for services such as mobile applications, video services, online social networking in cloud infrastructure. The author analyzed the characteristics of incoming requests and mechanism used in a cloud environment to characterize the workload.

3.2.2 Workload Prediction for Cloud Applications

The auto-scaling methods are grouped into two categories according to the elasticity techniques. The first category is reactive, where the scaling rules are set by the human expert or the customer. The second is proactive, it predicts the incoming workload from the historical component.

In literature, the vertical and horizontal scaling techniques are proposed using the reactive technique. Bonvin et al. [148] applied efforts to raise cloud data center profit and performance using a reactive approach. Yang et al. [27] developed a reactive approach to setup the data center resources based on the workload of the application hosted in the cluster. The users defined the threshold rules to make the scaling decisions. Zhang et al. [149] developed a reactive approach to classify the hybrid cloud workload into the trespassing workload and base workload. The prediction model ARIMA applied for scaling to the public cloud from local infrastructure.

The constraints of the reactive technique are that it responds to scaling only beyond the threshold values of throughput or resource utilization. It took more time to reconfiguration in case of the flash workload. The situation, when the additional resources are not available, the user will experience poor QoS. The preparation of the required resource before incoming actual workload is called the proactive auto-scaling technique. The prediction techniques help to develop the new proactive auto-scaling mechanisms.

Jiang et al. [150] developed a solution through prediction models. The problem of capacity planning and provisioning of resources could be fixed with the proposed approach. The resources must be prepared for the incoming workload in advance. Classical prediction methods of the time series were applied to forecast the workload of the web applications. Zhu et al. [61] developed a method to scale the VM type, VM cores, VM speed, and VM memory vertically using control theory. The ARMAX prediction

method used to predict an application CPU cycle and memory requirement.

The Exponential Smoothing (ES) and second-order ARMA prediction models used to analyze system behavior and time series prediction [68]. Roy et al. formed linear equation using the variables used for the application such as cost, QoS and resources. Messias et al. [123] applied the prediction method Naíve, exponential smoothing, AR, ARMA, ARIMA, etc. The Genetic Algorithm (GA) employed to select the prediction model. This technique provides the adequate performance along with certain overheads of GA. Calheiros et al. [122] used the ARIMA model for cloud applications workload forecasting and estimated the impact of the designed technique on the QoS parameter.

The machine learning approaches also applied in the literature to forecast the workload of the various applications. Sapankevych et al. [151] presented a survey on applications applied Support Vector Machine (SVM) for time series prediction. It found that SVM can predict the time series accurately with nonlinear and non-stationary data. The resource prediction applied for the multi-tier web application environment. Bankole et al. [152] applied Neural Network (NN), Linear Regression (LR) and SVM. It concluded that SVM is more accurate to predict than NN and LR. Islam et al. [125] applied LR and NN models to develop a resource assessment and provisioning model. Liu et al. [87] developed a auto-scaling based on pattern discrimination. The integer assignment problem used as per workload patterns for selecting SVM and LR prediction models. Performance prediction performed using [153] workload characteristics with the resource provisioning machine learning approach. Markov and Bayesian techniques were used to predict the workload for cloud-based applications.

In the present study, the researcher developed a short-term prediction model for web applications in a cloud environment. The classification approach selected the prediction models LR, ARIMA, and SVR based on workload pattern. The sliding window approach applied to consider the latest historical workload to maximize the accuracy and speed of the devised model. The primary goal of the study is to enhance the Quality of Experience (QoE) of the end users and reducing the resource oscillation.

3.3 System and Application Models

An elastic system of public clouds provides services to the end users with PaaS and SaaS layers. The PaaS layer can be the third-party providers interacting with the IaaS layer.

The SaaS providers concerned with web requests. The incoming request is processed on virtual machines of the IaaS layer. The primary purpose of this study is to make the efficient use of virtual resources with elasticity feature while meeting the Service Level Objectives (SLO).

The elastic system took the platform status from the monitoring phase and future incoming traffic status from workload forecaster. The forecasting module used historical data to generate the forecast that is subsequently sent to the auto-scaler for the decision to scale-in or scale-out. This resolution inputs resources management to act on/off the VM from the resource pool for virtual resources.

In the proactive elastic cloud system, one of the key components is the workload forecasting module. In this section, the researcher discussed the proposed model design with details of its input and output.

The research focused on the resource provisioning of the web application in the cloud. The web servers processed the HTTP requests from clients. The precise model of prediction can improve the QoE of end-users. The SLA is defined as T_s for processing user requests with an agreed rejection rate.

3.4 System Architecture

An elastic system is an important characteristic of cloud computing. It scaled the resources as per the user-defined application demand. The VM start and setup time is non-negotiable. Therefore, the client found the delay in QoS leads the penalty to the service provider.

Furthermore, the reserved resources can handle the flash crowd. But, it has been observed most of the time reserved resources remain underutilized. In addition, the reserved resources cannot handle the requests without the proper estimation. Thus, the situation leads to poor QoS to the end-user and application provider has to pay penalty.

In the present study, the researcher proposed the Technocrat ARIMA and SVR Model (TASM) prediction model. The designed model can effectively capture the workload pattern and find a suitable prediction model for different workload patterns in the sliding window. The proposed technocrat provisioning architecture is shown in Figure 3.1. The technocrat workload predictor implements the TASM prediction model in a cloud environment.

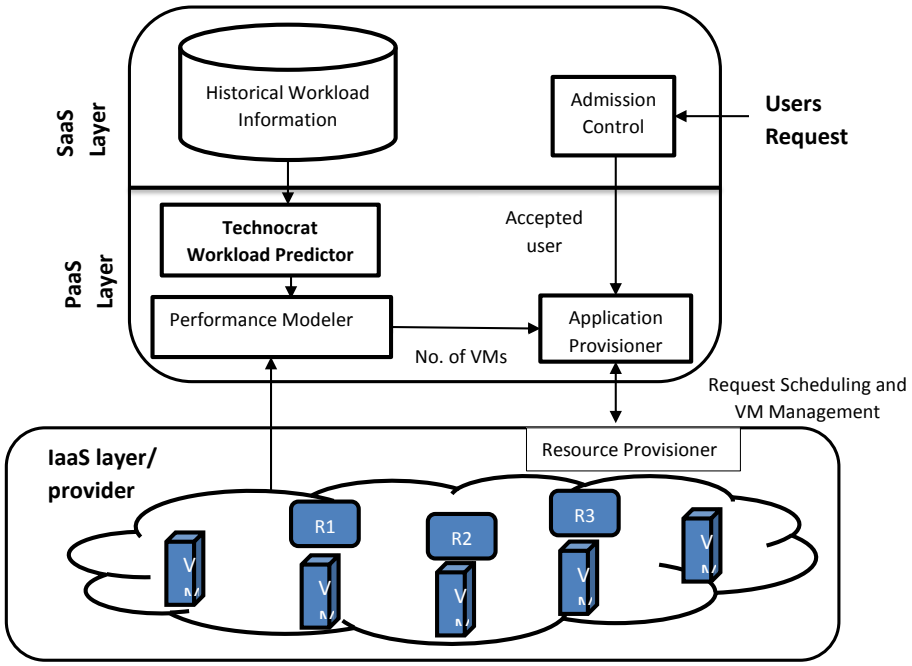

Figure 3.1: Technocrat cloud provisioning architecture

The main components of the cloud provisioning system is shown in Figure 3.1 are:

1. *Technocrat Workload Predictor:* The workload of the application predicted for the incoming traffic. The number of forecast requests passed to the *Performance Modeler* for the next time slot.

2. *Performance modeler:* The future workload estimate got from the *Technocrat Workload Predictor* module and estimate the VMs for the *Application Provisioner* module.

3. *Application provisioner:* The incoming requests input to the *Admission Control* component. It accepts or rejects the incoming request as per the present state of the system. The admitted user requests scheduled to the VMs having required capacity for processing. It also takes information about the estimated number of VMs from the *Performance Modeler*. If the number of the required VMs varies from actual VMs capacity, then the autonomous/automatic action taken by this module to scale up/out or scale down/in.

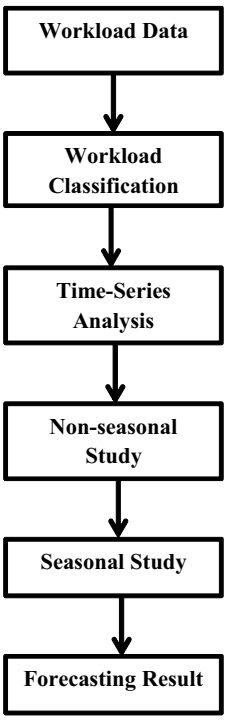

Figure 3.2: Generic forecasting methodology

A detailed application workload behavior is required to design an robust architecture. The log files moved to the central repository for *Historical Workload Information*. Further, these log files are used as a training set for the forecasting models to predict the future workload.

3.4.1 Technocrat Workload Predictor

Once the architecture is understood and components are defined, the researcher further discussed the workload predictor design. Figure 3.2 shows the general scheme of the forecasting process.

1. *Workload classification* module takes historical information on the workload and analyzes the pattern of workload. In addition, this module selects the appropriate model of prediction based on the pattern of workload. The pattern of workload classified according to change in velocity and intensity of incoming requests. The high time-scale workload characterized by non-linearity, while the low time-scale workload is linear in nature. Therefore, the peak-to-mean ratio varies in different workload patterns. In the high time-scale, the rate of change is higher as compared to the low time-scale workload.

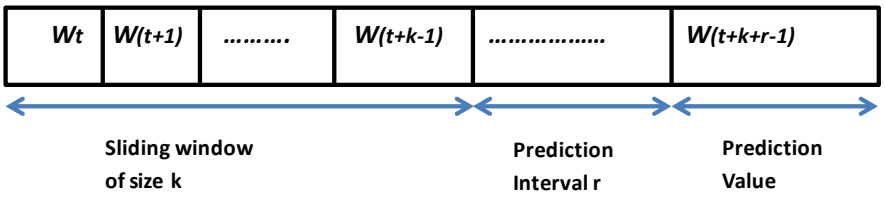

Figure 3.3: Sliding window approach for workload forecasting

2. *Workload predictor* designed to predict time series data using the LR, ARIMA and SVR models. A single forecasting model is unable to predict different types of patterns in the workload. ARIMA is employed for the prediction of the fast-scale workload and, SVR and LR models are deployed for the slow-scale workload patterns.

3.5 Research Methodology

In the present study, the researcher considered the proposed prediction model to be the agent and the task to be the workload. The problem is mounted as a problem with the assignment of the agent-task. The classification technique designed to choose the suitable model of prediction for different workload patterns.

3.5.1 Workload Classification

The startup time of resources (VMs) generally varies from 5-10 minutes in the elastic cloud system. Thus, the granularity of the time series prediction should be coarse-grained and latency period must be greater than or equal to the supply speed of VMs.

The researcher contemplated the model as a discrete time series. The proposed model time intervals split into frames. The $t \in \{1,....,K\}$ is real-time data frame in w_t sliding window. The sliding window captures the latest lags for the prediction model to takes the input of x sequences of k size for the output of w. Figure 3.3 shows the sliding window used in workload forecasting.

In this study, the researcher considered i types of incoming workload and, different web applications types described as $W_i(i = 1,...,n)$. Suppose j kind of forecasting models $P_j(j = 1,2)$ suits for various types of web applications workload W_i, as per

the workload pattern of the web applications. Allocate P_j to W_i, the t_{ij} represents the prediction time, and the prediction model j Mean Absolute Percentage Error (MAPE) $\varepsilon_{ij}(i = 1,...,n, j = 1,2,3)$ for incoming workload i generated at time frame t with the prediction model.

$$\varepsilon_{ij} = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{w_s - \hat{w}_s}{w_s}\right| \times 100\% \tag{3.1}$$

The w_s entitled the actual output, and $\hat{w}_s$ represents the prediction output at time interval s, n is the number of observations in dataset at time frame t.

The classification of web applications workload performed using rate-of-change of incoming workload as; fast time scale and slow time scale. The rate of change of fast time scale is relatively high with a comparison to the slow time scale in the time-series. The average rate of change in workload is defined as:

$$\overline{\omega} = \sum_{s=1}^{n}\left(w_{s+1} - w_s\right)/N \tag{3.2}$$

The time series with a fast scale have a greater sum of $l_2 - norm$ with a comparison to $\overline{\omega}$. The fast scale data is allocated to the ARIMA prediction model. The slow-scale workload will be assigned to the SVR prediction model. The ARIMA gave higher accuracy for fast scale series while SVR has better accuracy in slow scale. The LR prediction model applied in two situations. First, the linear time series or bad results found during evaluation in the other prediction models. The fast prediction speed observed in LR as compare to other prediction approaches. The ARIMA lag values will be used in the LR model if ARIMA gives a poor result for fast scale data.

The model uses $l_2 - norm$, an average rate-of-change and error residuals to generalize the classification problem. Usually, web application jobs are small to medium size with an inconsistent workload pattern.

3.5.2 Workload Classification Model

A heuristic approach designed to classify the workload. The heuristics designed according to the residual test, Average Rate of Change (ARC), linearity test and $l_2 - norm$. According to SLA, the cumulative platform error is allowed. The $E = (E_1,...,E_i,...,E_n)$ defined the platform error. The controlled decision vector v_{ij} described the prediction

model from the sliding window for the newly added frame after each frame has been executed. The research goal is to minimize the prediction error after selecting v_{ij}.

$$min \sum_{i=1}^{n} \sum_{j=1}^{2} e_{ij} v_{ij} \quad 0 > \varepsilon_{ij} < E_{ij} \tag{3.3}$$

The model of classification generalizes the problem with the average rate-of-change and $l_2 - norm$ of the workload, further linearity and residual help in model selection. The daily, monthly or weekly pattern dominates the web applications workload.

The linearity of the original series analyzed to perform a linear series test. This procedure is recommended as the non-linear model is usually over-fitted with linear series or weak fitting achieved with flexibility.

In this model, researchers used *teravirta.test* from the *tseries* package in *R*. This library contained the implementation of *Teravirta* [154], a non-linearity test.

In case, the time series found non-linear ARIMA applied to the time-series. If the evaluation results are unsatisfactory, then use an LR method to improve the achieved results. ARIMA results from the non-seasonal study can be used to devise the models by incorporating the significant lags.

The lagged values derived from the ARIMA results. Afterward, build the models using the significant (non-seasonal) lags extracted from the analysis to create an embedding scheme of lags.

Finally, the researcher performed the evaluation criteria again to examine the quality of the new models.

3.5.3 Forecasting Models

In this section, the researcher explained the prediction models used to developed the technocrat forecasting model.

ARIMA model is a joint of ARMA with differencing. The model defined as *ARIMA* (p, d, q); the p represents the lagged values under consideration of autoregressive section, d is the differences and q denotes the lagged values. The implementation of this model in *R* [155] is known as Hyndman-khandakar algorithm [156] is present in $forecast$ package.

Support Vector Regression (SVR) model is data analysis for classification and regression analysis. It is a supervised learning model [157]. The main purpose is to solve

quadratic optimization problem in high dimension to discover the separability hyperplane by space [158]. In experiments, parameter settings of SVR: $c = 100, m = 3, g = 0.5, p = 0.001, t = 2, s = 3, epsilon = 0.1$.

Linear Regression (LR) model considers the regression model $E(Y|X)$, where Y is a response variable and $X_1, ..., X_p$ denotes the input. It is a process of analyzing how the output is affected by the input [159]. It is one of the simple prediction technique. The linear model is among the fast prediction model, but it has the issue of non-linearity prediction [160]. The researcher applied an LR method in the proposed prediction model for linear historical series, or the non-linear model gives unsatisfactory results. In *R* tool, a *lm* method used for the prediction using the LR method.

3.5.4 Time Series Analysis

The workload is available in HTTP log format from the web trace files. The researcher generates the discrete time series from the HTTP weblogs.

Data analysis in time series forecasting is critical. Here, the researcher manually examine the ACF and PACF data to configure the forecasting models. The plots of ACF and PACF provides seasonality, trends, and stationarity information. Figure 3.4 shows an analytical forecasting technique. The $diff$ function in the time series is applied, if the time series is non-stationary. Time series preprocessing performed to remove the time series seasonality. The classical approach applied to the time series to identify the series trends, seasonality, and irregularity.

The analytical process is explained in the steps. Firstly, ACF and PACF plot produced to analyze the significant lags manually. In case, there is no significant lag present in the series, the maximum 5 lag value considered for the non-seasonal pattern and seasonal pattern took 144 lags for 10 minutes discrete model (1 day). The researcher considered a maximum of 5 lag values for non-seasonal due to the possibility to depict auto-correlation from the first 5 lags in most of the time series. The technique of time series decomposition used to determine the seasonality, irregularity, and trend in the series. As observed from the current position, the sliding window provides the last five values. Seasonality is removed from the historical workload frame by referring to the model of decomposition generated during data pre-processing.

A classification approach performed on the values of the sliding window to find

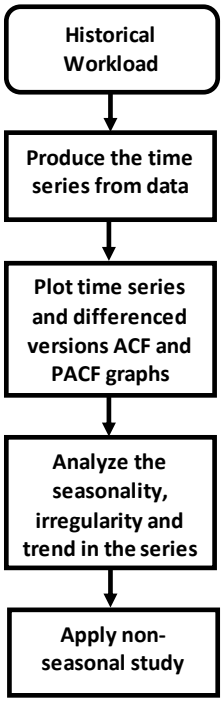

Figure 3.4: Analytical process of forecasting model

the appropriate prediction model. The suggested prediction model further applied to study the incoming workload pattern of the seasonal and non-seasonal workload. It has decided that a seasonal study would be for 144 lags (1 day) because the website usage pattern matches weekdays. The observation of the last 5 values from the sliding window entered into the classification model suggesting the appropriate prediction model for the present frame to predict the upcoming value in the time series. If seasonal patterns are discovered in preprocessing, then additional classical multiplicative time series model will be applied to the non-seasonal forecast result. Although, the models are flexible to select the seasonal model, even then most of the models cannot find the seasonality. To calculate the residuals, the evaluation criteria have been applied. In addition, the sliding window provides recent history lag values. This process is continued for every 10 minutes, and gives the incoming workload value.

3.5.5 Non-seasonal Study

The investigator analyzed the series and designed the model using LR, ARIMA, and SVR as shown in Figure 3.5. The selection of the appropriate prediction model with the proposed classification model. The sliding window observations are classified as per the

pattern and further select a suitable model of prediction automatically. If the observations are of slow scale, then LR or SVR model found a suitable for the particular series. The predicted value passes through the evaluation criteria. The workload forecasting performed in *R*, Support Vector Regression (SVR) implemented in *svm* function from the *e*1071 package and *lm* for linear regression.

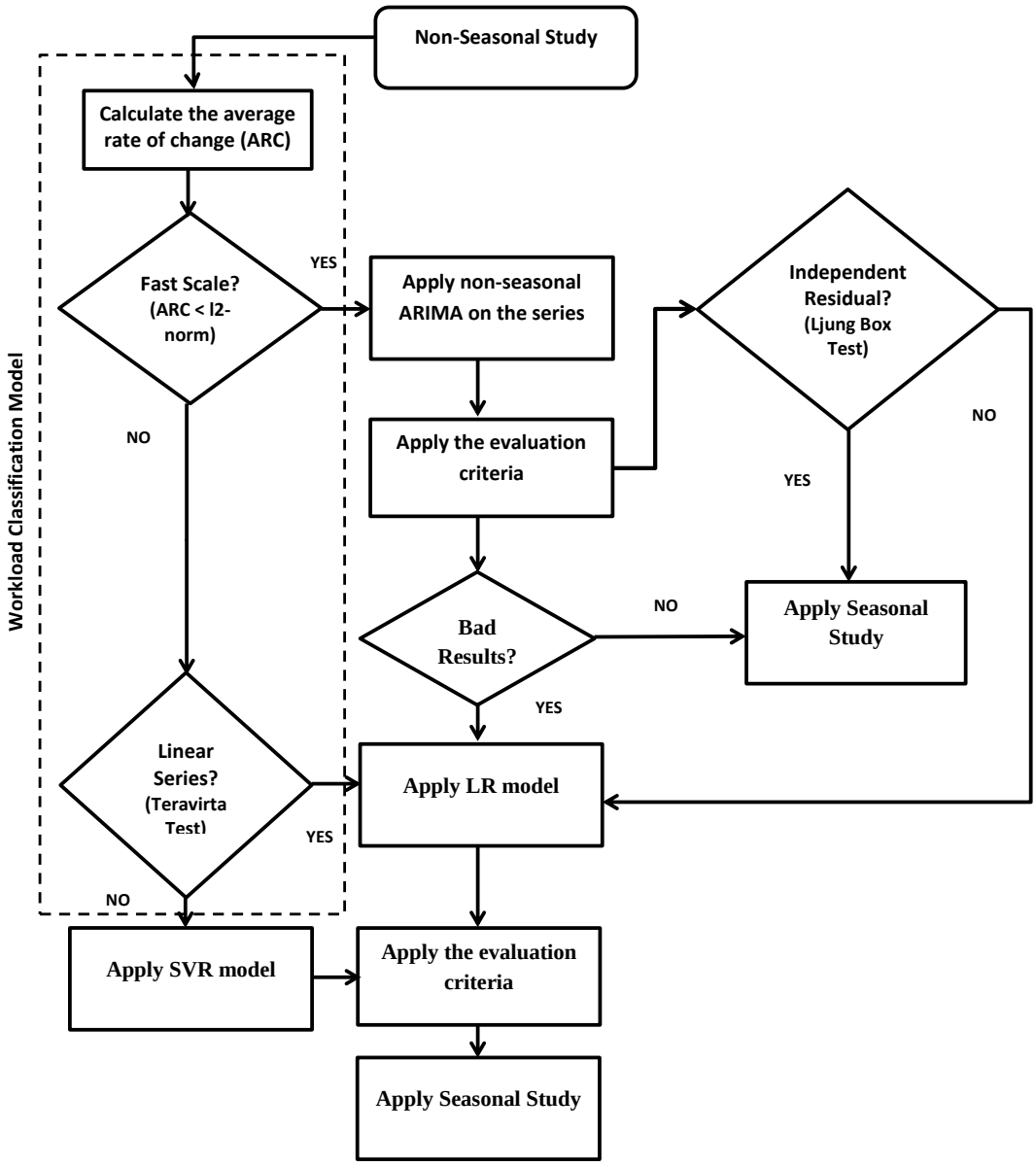

Figure 3.5: Non-seasonal forecasting model

The ARIMA model parameters can be generated with the function *auto.arima* in R. This procedure is based on the observations of AICc. Further, the investigator incorpo-

rated the examine criteria for accessing the quality of the model.

Afterward, analyzed the residual independence from Box-Pierce [161] and Lung-Box [162] tests. It analyzed whether or not the white noise process differs substantially from h auto-correlations. If residual still present in the auto-correlation, then continue to explore the model.

Furthermore, the researcher checked the series with models of slow scale like *LR* and SVR. The popular non-linearity test named as $teravirta.test$ is present in $tseries$ package of *R*. If the series is found linear, then the *LR* model applied to predict the series otherwise SVR used to forecast the time-series.

3.5.6 Seasonal Study

The series may contain the daily seasonality. In the present study, the seasonality in the series found from historical workload using the analytical process shown in Figure 3.4. The classical approach of the time series used to decompose the series. This seasonal pattern used to improve the outcome with seasonality in the *LR* and SVR models. In addition, the model is given a flexible approach to select the seasonal order from the sliding window for the given data.

The seasonal model and differentiated value can be evaluated using *ARIMA* model. Ljung-Box test applied as a tool for evaluation criteria for comparing the new model with existing models. If the null hypothesis is rejected, the model selects the *LR* to improve the outcome and consider both the final forecast value outcomes. In order to obtain analysis for new models, *LR* uses *ARIMA* information of lag values. The repeated prediction values obtained from *ARIMA* model, the final values obtained after the evaluation of comparison results.

3.6 Experiment and Analysis

In this section, the researcher describes the experiment setup, dataset details and pre-processing of dataset. The researcher also discussed the parameters configuration. Two real time datasets collected for the performance evaluation of existing and proposed time series prediction model. The preprocessing of dataset is performed to ready the weblogs in discrete series. Further, prediction accuracy of forecasting models tested using the statistical programming language.

Table 3.1: Summary of input workload

Dataset Name	**ClarkNet**	**NASA**
File name	Aug28log, access1	accesslogAug95
File size	171.0 MB, 172.5 MB	167.8 MB
Length (s)	3,328,587 (14 days)	3,461,612 (28 days)
Timestamp resolution	1 second	1 second
Sampling interval	10 minutes	10 minutes
Number of Samples	2016 (14 days)	4032 (28 days)

3.6.1 Dataset

The web applications datasets emulated for the experiment are ClarkNet [163] and NASA [164]. The details of the datasets are described in Table 3.1.

Metro Baltimore-Washington DC area WWW server log collected for ClarkNet series. It has the HTTP requests of the WWW server of 2 weeks traces. The first week log collected from 28 August, 1995, to 3 September, 1995. The second-week log obtained from 4 September, 1995, to 10 September, 1995. The total number of requests observed in the duration of 2 weeks are 3,328,587.

The 2 months WWW server log is collected from NASA Kennedy Space Center Florida. The duration of the log is from 1 July, 1995, to 31 July, 1995, and second log is from 1 August, 1995, to 31 August, 1995. The total 3461612 requests recorded in two months.

3.6.2 Accuracy of Prediction Models

The performance metrics are Mean Square Error (MSE); Root Mean Squared Error (RMSE); Mean Absolute Error (MAE); Mean Absolute Percentage Error (MAPE) used to evaluate the proposed model.

$$MSE = \frac{1}{n}\sum_{s=1}^{n}(w_s - \hat{w}_s)^2 \tag{3.4}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{s=1}^{n}(w_s - \hat{w}_s)^2} \tag{3.5}$$

$$MAE = \frac{\sum_{s=1}^{n}|w_s - \hat{w}_s|}{n} \tag{3.6}$$

$$MAPE = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{w_s - \hat{w}_s}{w_s}\right| \times 100\% \tag{3.7}$$

The minimum value of error metrics concluded the best model. The n parameter has denoted the number of observation in the past time series. The current workload in the frame is denoted with w_s and the predicted workload represented with $\hat{w}_s$.

3.6.3 Accuracy of Auto-Scaling

The resource allocation calculated using the $M/M/m$ queuing model principle. In order to calculate over-provisioning and under-provisioning, the platform error is evaluated at a given point of time. The usage of the system is described according to the [155] model.

$$\rho = \frac{\lambda}{m\mu} \tag{3.8}$$

Eq. 3.8, ρ is the system usage, λ is the arrival rate, m is the number of servers required and μ is the processing rate. The ideal ρ case is less than 1 to maintain the stability of the system. The goal is to maintain stability at less than m. The Equation 3.8 can write as follows:

$$m = \left\lceil \frac{\lambda}{\rho\mu} \right\rceil \tag{3.9}$$

The response time required to calculate the desired value of ρ as per SLA and the value of μ as processing time. The response time can be calculated as follows [155]:

$$R = \frac{\frac{1}{\mu}}{1-\rho} \tag{3.10}$$

After putting Eq. 3.10 in ρ function 3.8 and gained the Equation 3.11.

$$\rho = 1 - \frac{1}{R\mu} \tag{3.11}$$

R denotes the client's assured response time as per SLA. The combination of Eq. 3.9 and 3.11:

$$m = \left\lceil \frac{R\lambda}{R\mu - 1} \right\rceil \tag{3.12}$$

As per the scenario, the parameters were replaced in the next time frame to predict the VMs [123]. The total resource oscillation calculated as follows:

$$Total_{osc} = \sum_{s=1}^{n} \left| \frac{R\hat{w}_s}{R\mu - 1} - \frac{Rw_s}{R\mu - 1} \right| \tag{3.13}$$

Oscillation is the sum of resource under-supply and over-supply. The workload forecast is $\hat{w}$ and at the given time s represents the actual workload as w. The n in the time series denotes the number of frames.

3.6.4 Experimental Setup

The implementation of the presented models performed in *R* tool. The parameters configured for most of the models. The ARIMA model has the feature to fit the parameters automatically up to lag 5.

In SVM [165], there is no systematic way to select parameters. The parameters of the SVR model are epsilon, cost, and gamma. Combination of parameters found using experimentation on a similar dataset small sample.

Table 3.2: Parameters grid

Model	**Parameters**
Naïve	–
LR	–
AR(p)	$p = 2$
MA(q)	$q = 3$
ARMA(p,q)	$p = 2, q = 3$
ARIMA(p,d,q)	*automatic*
SVR	m = 3, c = 100, g = 0.5, s = 3, p = 0.001, t = 2, epsilon = 0.1

In addition, all methods of forecasting with different parameters are trained on the training dataset. The 80% data is used for the training purpose. However, the designed technique could work in real time environment without any training. The selected models are displayed in Table 3.2 on the test dataset with parameter configuration. The 20% time series data used for testing purposes.

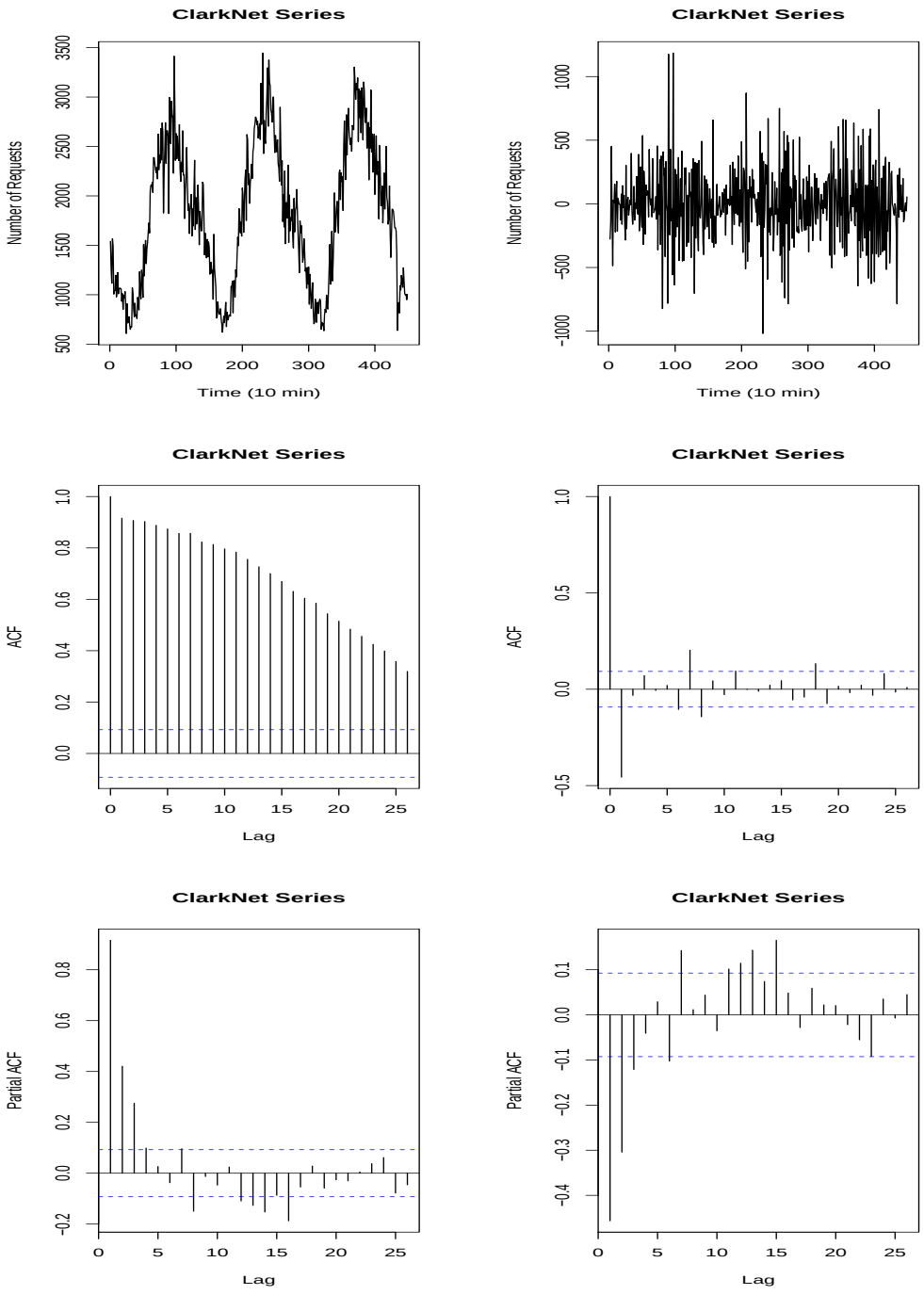

Figure 3.6: ClarkNet 10 minutes time series, ACF and PACF plots

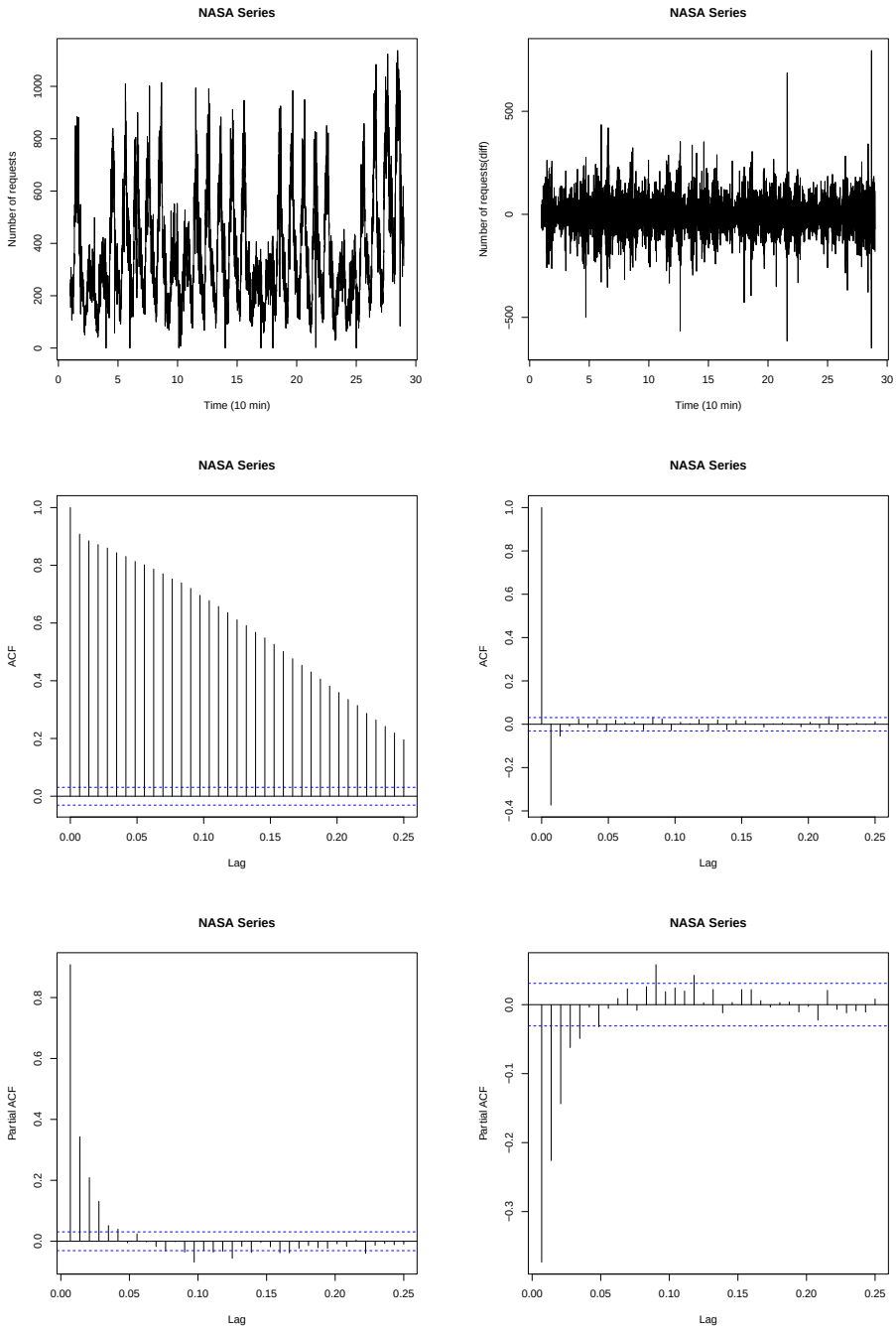

Figure 3.7: NASA 10 minutes time series, ACF and PACF plots

3.6.5 Time Series Analysis

In the present study, the researcher designed a forecasting model for short-term prediction. The discrete series generated from the workload of time series during the lag of every 10 minutes. Researchers observed the daily seasonal pattern in the workload time series. Such a pattern is not reflected in the ACF and PACF plots. The series of indifference has a significant lag in ACF showed 12 and PACF showed 16 respectively

shown in Figure 3.6.

In the present study, the seasonality is observed from the 144 lag values with each lag of 10 minutes to sense the seasonality in last 24 hours. The sliding window time frame picked and applied the classification approach. In the case of non-seasonal study, the researcher used the 5 lag values of 10 minutes series mostly used in time series prediction in literature. Auto-correlation can usually be observed from the first lags of 5.

The results showed that there is no seasonality in the ClarkNet and NASA series as shown in Figure 3.6 and Figure 3.7 respectively. This could be analyzed from the ACF and PACF plots. Thus, the researcher continued to apply the non-seasonal study. As the proposed approach is towards the autonomous system, pre-processing was done before the non-seasonal study series. The classical multiplicative time series model used to decompose the series according to the analytical approach.

Non-seasonal Study

The model was used SVR, ARIMA, and LR to build the non-seasonal prediction model. The proposed classification approach applied as per sliding window approach on each time frame in the first phase. The adequate model selected in the original time series through the classification approach. The parameter configuration is not required by the ARIMA method in R. This can be performed with the $auto.arima$ automated version. The ARIMA function used in this process with the seasonality restriction. The configuration of the SVR model parameters as follows: $s = 3, p = 0.001, m = 3, c = 100, g = 0.5, epsilon = 0.1, t = 2$.

The experiment results of ClarkNet Series in Table 3.3 showed that the *AR* model gives the best results in existing models. In the case of the NASA series experiment, the best results are obtained from existing models is *LR* Table3.4. Therefore, in general, no predictive model is best. Compared to other models, the proposed model in the present study performed significantly better. The classification approach helps with different workload patterns to select the appropriate model. In the case of ClarkNet series, the accuracy of SVR is unsatisfactory, and ARMA delivers the poor results for the NASA series. Figure 3.8 and Figure 3.9 indicate the proposed model prediction efficiency. It has been observed that the proposed model able to predict the workload more accurate

Table 3.3: Experiment results of non-seasonal study of ClarkNet Series (5 orders)

Model	MAE	MSE	RMSE	MAPE
Naïve	192.75	65561.45	256.04	12.87
LR	207.63	79569.25	282.08	14.52
AR	178.81	57862.46	240.54	12.41
MA	197.15	69282.43	263.21	13.78
ARMA	219.24	81878.24	286.14	15.19
ARIMA	181.19	58415.94	241.69	12.47
SVR	230.65	93023.06	304.99	16.12
TASM	152.40	45911.11	214.26	10.72

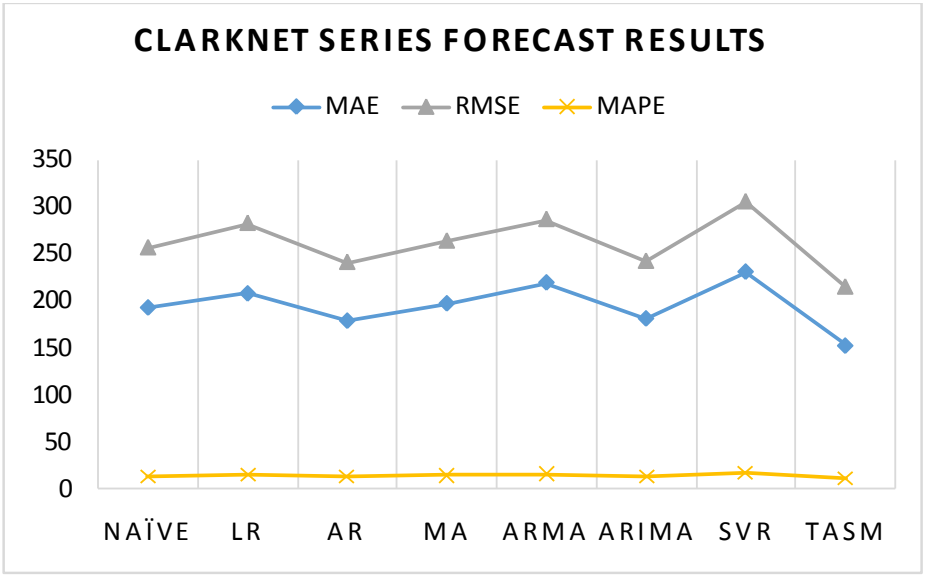

Figure 3.8: ClarkNet series forecast results

Table 3.4: Experiment results of non-seasonal study of NASA Series (5 orders)

Model	MAE	MSE	RMSE	MAPE
Naïve	70.02	9330.15	96.59	19.67
LR	50.81	5427.85	73.67	14.67
AR	66.95	8066.15	89.81	19.42
MA	66.20	7876.31	88.75	19.26
ARMA	80.36	12476.14	111.70	22.88
ARIMA	66.39	8028.82	89.60	19.19
SVR	66.77	8617.73	92.83	18.79
TASM	47.76	4891.24	69.94	13.76

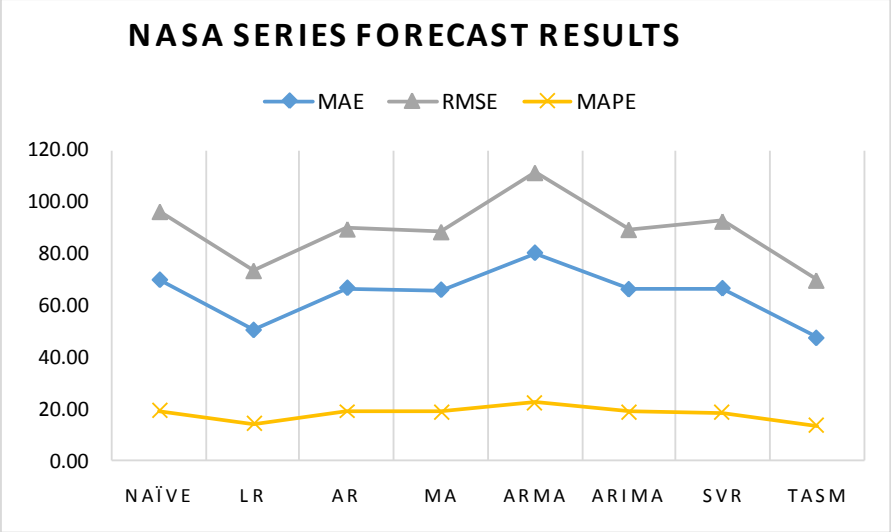

Figure 3.9: NASA series forecast results

as compared to other prediction models.

Seasonal Study

The seasonality could be present in the time series. The seasonality can be observed in the testing phase. The researcher applied data preprocessing and further applied the classic multiplicative model to remove the seasonality. To improve the series result, an additional seasonal pattern is used. In the 10 minutes series, the daily seasonality means 144 lag values. ARIMA model can automatically select the seasonal order, *LR* and *SVR* model are configured manually. Once the data input into the model, the seasonal pattern recorded and the result improved with seasonality. There is no seasonality present in this series. However, having the same outcome as the non-seasonal study. The order $1,0,1$ for selected automatically, when applied $auto.arima$ on the entire series.

Accuracy of Resource Provisioning

The over-provisioning and under-provisioning of the resource provisioning techniques are calculated using Eq. 3.10 and Eq. 3.11. The performance result obtained from the experiments mentioned in Table 3.5 and Table 3.6 for ClarkNet and NASA time series in terms of platform resource allocation. In the experiments, the following scenario is considered: response time as per SLA is $0.4s$ and processing rate (μ) is 1 request per

Table 3.5: Experiment results of ClarkNet series resource provisioning

Model	Under-provisioning	Over-provisioning	Total
Naïve	151	129	280
LR	172	139	311
AR	148	127	275
MA	168	135	303
ARMA	180	136	316
ARIMA	151	129	280
SVR	182	131	313
TASM	128	116	244

second. In the case of ClarkNet series, AR gives better results where LR is adequate for NASA series as compared to existing models. The proposed model TASM performed fairly well in saving resources with QoS to the end-users. The performance of the Naíve model is also good, which is the basic prediction model. The comparison results of resource oscillation are shown in Figure 3.10 and Figure 3.11 for the ClarkNet and NASA time series respectively. The predictive accuracy of the TASM prediction model for ClarkNet and NASA workload shown in Figure 3.12 and Figure 3.13.

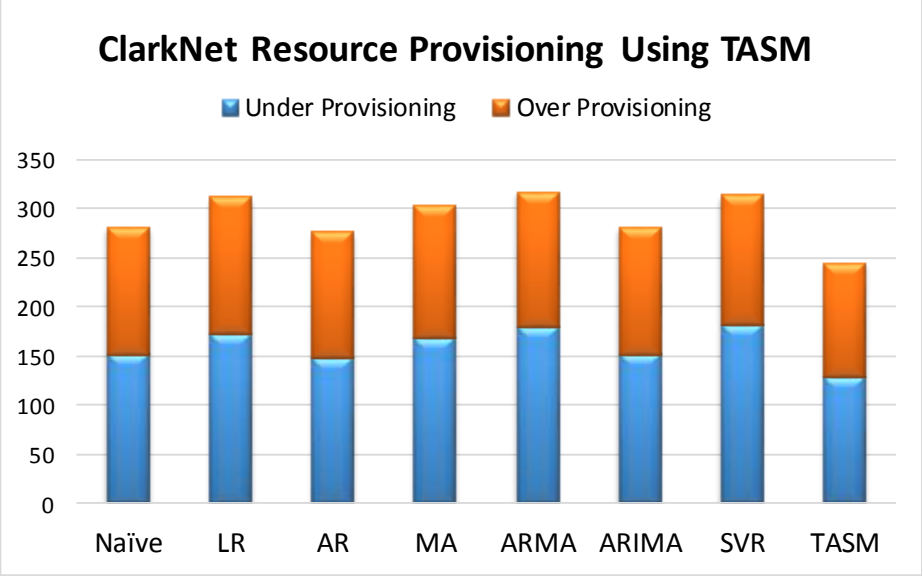

Figure 3.10: ClarkNet series resource allocation

Table 3.6: Experiment results of NASA Series resource allocation

Model	Under-provisioning	Over-provisioning	Total
Naïve	117	112	229
LR	77	85	162
AR	108	96	204
MA	108	94	202
ARMA	140	127	267
ARIMA	110	95	205
SVR	112	108	220
TASM	77	73	150

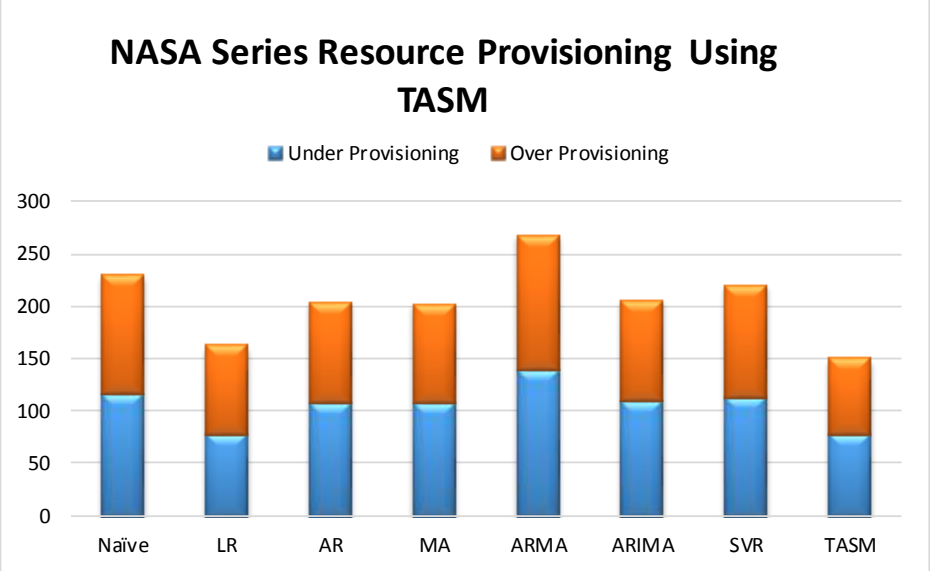

Figure 3.11: NASA series resource allocation

The proposed model can be an effective tool for the proactive auto-scaling technique. The reason being the web application have irregular request pattern and most of the pattern is nonlinear in the workload. The total time-series depicted the high scale and non-linear series, but sliding window contains some linear and slow scale frames. So, it is recommended to use the proposed model in resource provisioning of web applications with mixed request patterns.

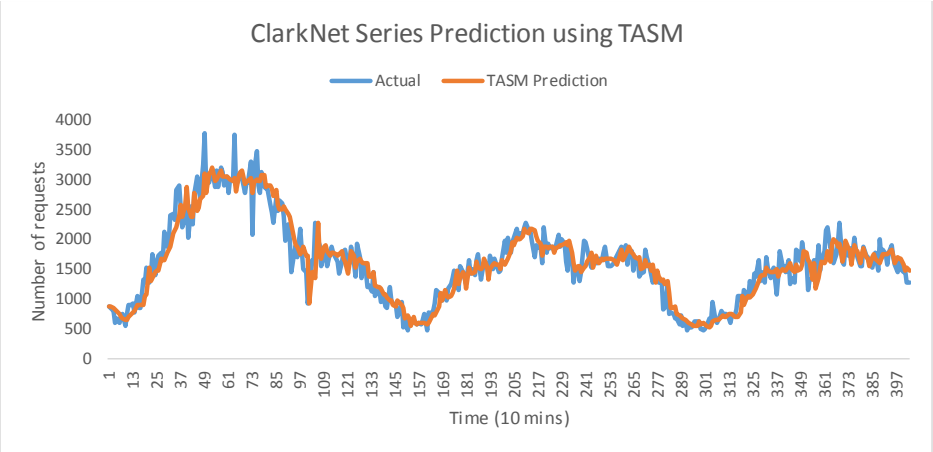

Figure 3.12: ClarkNet Series prediction using TASM

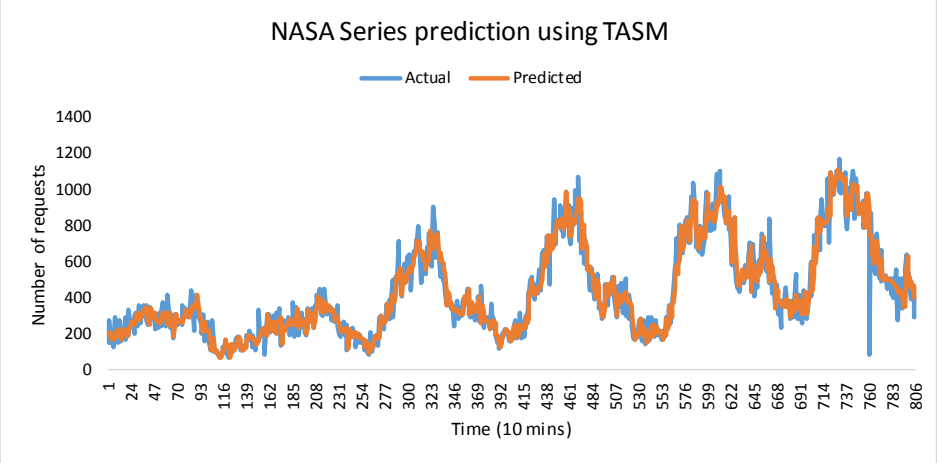

Figure 3.13: NASA Series prediction using TASM

In experimental results, the proposed model shows the under-provisioning and over-provisioning of the resources are minimum as compare to the state-of-the-art prediction models. Thus, it can provide the benefit in terms of cost to the customers adopts on-demand pricing policies in cloud infrastructure.

3.7 Summary

In this chapter, cloud provisioning architecture discussed with a new prediction module called as a technocrat workload predictor. The new model designed named as Technocrat ARIMA and SVR Model (TASM). VM startup takes 5 to 10 minutes in the cloud environment. Thus, the workload prediction could be a solution at the primary stage,

which helps to prepare the resources for the incoming workload in cloud data centers.

The workload of the web application is a mixture of different patterns of time series. There is no general model that fits all types of time series in the literature. In the present study, the researcher proposed a classification-based workload prediction model TASM, which provides a better understanding of the workload and helps to select the prediction model. Furthermore, the prediction model analyzed residual testing for model parameter configuration. The TASM model has effectively predicted both seasonal and non-seasonal patterns.

The accuracy of the model was evaluated with metrics MAE, MSE, RMSE, and MAPE. In addition, platform error was calculated as over-provisioned and under provisioned resources. The experiment results showed that the proposed model helps to strengthen the auto-scaling and provisioning mechanism for web applications in the cloud environment.

CHAPTER 4

A ROBUST HYBRID AUTO-SCALING TECHNIQUE FOR WEB APPLICATIONS IN CLOUD

4.1 Introduction

According to recent trends in the age of cloud computing, various application providers host cloud applications rather than purchasing computer infrastructure. The resources are offered by cloud providers like Amazon EC2 with additional scalability and pay-per-use model features in the form of virtual machines for Application Providers (APs) [31, 166, 49]. Three pricing models are offered by the cloud providers named as *reservation*, *spot* instances and $on-demand$. Virtual Machines (VMs) are supplied on-demand and can be purchased at a fixed price on an hourly basis. In the reserve VMs, the APs must set the contractual period and prices for a number of VMs. Amazon EC2 launched the spot pricing policies for unused capacity.

Amazon sales unused VMs capacity through an open market bidding mechanism. A bid is made through an auction mechanism to define the maximum unit price for the VM number and type in the cloud. The resources will be allocated to the cloud user if the offer price is higher than the current place price. Spot cases can be interrupted due

to the number of reasons, including an increase in spot prices over the maximum price, capacity can no longer be provided, or increase in the demand for spot instances [167].

Spot instances are cost-effective and can be interrupted for non-time-critical applications. The constraints in sport instances are availability and time restriction, which made it a non-suitable pricing policy for web applications.

The web application providers is concerned about the web environment's dynamic characteristics and incoming requests patterns from the end users. Therefore, the static supply of resources is not an effective technique. As the incoming user request rate increases also generate the under-provisioning condition. This causes the user requests to be delayed or interrupted. In the second scenario, when traffic is reduced cause the over-provisioning situation which results in higher renting costs to the APs [49, 74].

The minimum amount of resources is prepaying by APs in order to obtain the discount on the rental when considering the different cloud pricing models (e.g. EC2 reserved rental receives up to 75 percent discount) [49, 168]. Afterward, APs prefer to use the short-term lease model to satisfy temporary needs with varying loads. This method is a highly efficient mechanism to determine capacity and on-demand rental resources according to the application workload [169].

In this study, the researcher designed a Robust Hybrid Auto-scaler RHAS) for web applications in the cloud environment. The technique is designed carefully to save costs together with Quality of Service (QoS). The MAPE cycle has important functionalities for the implementation of an autonomous system and the saving of rental costs. The present study put more focus on the analysis and planning phase of the MAPE architecture to ensure strengthen the auto-scaling decision making process. The MAPE loop features are used for the auto-scaling process.

The main contribution of the present study is as follows:

- Designed and developed a hybrid analysis method for the auto-scaling of resources for web applications.
- Designed and developed a hybrid planning mechanism for the scaling decisions in the cloud environment.
- The performance evaluation of the proposed approach carried under real-world workload traces for different metrics.

4.2 Background

Auto-scaling is a technique to dynamically adjusts the resources allocated to elastic applications as per the incoming workloads. Auto-scaler in the cloud environment is generic while some are application specific to meet the Service Level Agreement (SLA), QoS and minimizing the renting cost. The auto-scaling challenge for the web applications is to dynamically grow or shrink the resources to meet fluctuated workload requirement. Autonomous scaling techniques work without human intervention. Autonomic systems are self (configuring-optimizing-protecting-healing) [23]. The auto-scaling following the MAPE-K loop: Monitoring (M), Analysis (A), Planning (P) and Execution (E), knowledge(k) [170, 24] shown in Figure 4.1.

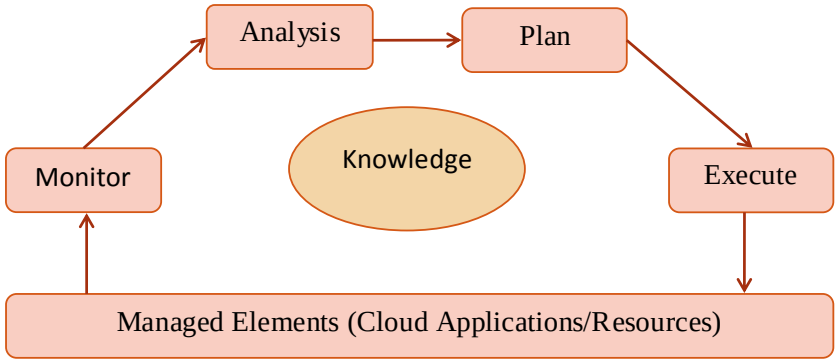

Figure 4.1: MAPE-K loop

1. **Monitoring**: The monitoring system collects the information from a cloud environment about the compliance of user expectations, resource status, and SLA violation. It provides the state of infrastructure to the cloud provider, and users get to know about application status with expected SLA. Auto-scaling protocols are decided on the basis of performance metrics for web applications. The author suggested parameters such as resize numbers, operating interval, decision duration, decision threshold, refractory period and instance bounds [25]. Generally, metrics provided by cloud providers are related to VM management, otherwise, it will be taken from the operating system. The proxy metrics are used to reduce the complexity of metrics such as hypervisor level and application level (e.g., CPU utilization, workload).

2. **Analysis**: The collected information is further processed in the analysis phase. It gathers all information from metrics and current system utilization and prediction information of future workload. Some auto-scaler are working on a reactive approach. The decision is taken after analyzing the current system state. The threshold values are fixed to scale in/out decisions, while others are using a reactive approach or both. Reactive is a sophisticated approach because it's always a delay between the settings of resources for scaling decision. The VM startup time varies from 350 to 400 seconds [26]. Flash crowd and events are still a challenge with the reactive approach.

3. **Planning**: Analysis phase evaluates the present state, now the planning phase has to decide to scale up/down or scale in/out to compliance with SLA and profit trade-off.

4. **Execution**: Execution phase is already decided in the planning phase. Cloud providers API is responsible for the execution of planning. The client is unaware of the issues in the execution phase. VMs are available to users for a certain period, the startup time of VM takes some time, and these delays have been already discussed with the user in resource SLA.

5. **Knowledge**: The knowledge to be shared among above all four functions stored in this repository. The shared knowledge contains metrics, historical logs, topology information, and policies. The information passed to the autonomic manager.

4.3 Related Work

4.3.1 Auto-Scaling Using Queuing Model

In the literature, the authors modeled the cloud servers as a queuing system. The queuing model $M/G/m/m+r$ is modeled the probability distribution for the response time in the cloud infrastructure [171]. In this article, the incoming requests inter-arrival time assumed to be exponentially distributed. Yang et al. [172] modeled the multi-server with $GIx/M/S/N$ queuing model. The general distribution of inter-arrival time, server times with exponential distribution, service capacity as finite with batch processes. The incoming request segregated into different subtasks, the performance is assessed with

the proposed queuing model. Urgaonkar et al. [92] calculated the required servers with Little's law and G/G/1. Further errors are removed with a reactive approach. Zhang et al. [93] applied regression model and QM to calculate the CPU demands in the future. Villela et al. [96] modeled the e-commerce applications with M/GI/1/PS queuing model. The characterization of incoming requests performed with the real traces of e-commerce applications.

4.3.2 Auto-Scaling Using Proactive Model

In the literature, the researchers mostly applied time series prediction model to forecast the multi-tier web applications workload. A Simple Moving Average (SMA) [115] gave less accuracy to predict fluctuating workload. Some authors applied MA to reduce the noise in time series [59, 60]. Huang et al. [116] proposed a resource prediction model with Double Exponential Smoothing (DES) and, the comparison performed with the Weighted Moving Average (WMA) and mean. The history records w helps to get better results in ES. Mi et al. [117] applied Brown's double ES to predict the workload and achieve a good result for HTTP workload with a small error. Aslanpour et al. [131] applied DES and WMA for the prediction of time series.

The auto-regression technique has also applied for workload and resource prediction [118, 119, 115, 30, 68]. Roy et al. [68] used the AR model for workload forecasting by taking the previous three observations. Further response time estimated from the predicted values. An optimization controller applied to find resource allocation, considering SLA violation cost, reconfiguration, and leasing resources.

ARMA model is a simple and efficient model to predict future workload (number of requests). Fang et al. [120] predict VMs CPU usage. ARIMA model is applied in various articles [121, 122, 123]. ARIMA required historical workload. The performance of the model highly depends upon the history window. ARIMA approach is ideal for dynamic workload such as web applications. Sedaghat et al. [121] applied the horizontal and vertical scaling to increase the benefit in terms of cost. Mao and Humphrey [124] used the classification given as increasing, stable, seasonal and on/off. Calheiros et al. [122] used ARIMA model for workload prediction, and evaluate the impact on different QoS parameters. The web application workload is dynamic and contains seasonal data. The model gives 91% accuracy for non-seasonal data, but not fit for the highly

non-seasonal workload. This work can be further extend using an adaptive approach for classification of workload, and design the heuristic for ARIMA fit function for different classes. As discussed earlier, one model doesn't fit for all types of workload, Messias et al. [123] present GA based approach for time series prediction. Traces of real workload have been used to evaluate the prediction model. A new metric has been introduced in the article named as an Elasticity Index (EI), which describe the solution optimization. The range of EI varies from (0 to 1), a value near to 1 means the solution is good. The model gives less error as compared to other models.

The accuracy of neural network [125, 126] and multiple regression equation [65, 126, 30] model are highly dependent upon on the size of input the history window. Islam et al. [125] used more than one value from the history and got a better result. Kupferman et al. [30] devised the necessity of balanced size of input history window. Regression of various window sizes applied to find the prediction values. The prediction interval of r is also an important factor. Islam et al. [125] investigate the size of the interval window and found 12 minutes an appropriate time, because of VM startup time is between $5-15$ minutes. Prodan and Nae [126] applied the neural network to forecast the game load for 2 minutes. In contrast, the neural network is better than MA and ES in terms of accuracy.

Time series analysis techniques are able to forecast the future workload of web applications. Further, this information can be used to predict resource requirements. The technique is very appealing because of input workload is known to the auto-scaler in advance, and have enough time to prepare the VMs beforehand. The drawback of techniques is the accuracy, which depends upon the input workload, history window selection, metrics, prediction interval, and target application. There is no best solution for all types of time series forecasting. In this study, the researcher has developed a robust auto-scaling technique with a hybrid approach. The analysis and planning phase carefully designed with classification based prediction model TASM and reactive technique to give QoS while saving the cost for application provider.

4.4 Proposed Approach

In this section, the researcher discussed the proposed approach for auto-scaling of web applications. The MAPE loop is applied for Robust Hybrid Auto-Scaler (RHAS) de-

sign. This approach is intended to estimate the necessary resources for the incoming workload in horizontal scaling. Table 4.1 describes the symbols used in this section.

Table 4.1: Notations used in the RHAS approach.

Component	Description
A_t^{rt}	Analyzed response time in last minute
A_t^{u}	Analyzed CPU utilization during $\triangle s$
$Clock$	Simulation timer
D	Scaling decision (e.g. ScaleUp, ScaleDown, DoNothing)
k	Size of sliding window
M_t^{rt}	Average response time at time t
M_t^{u}	Average CPU utilization at time t
RT^{lowThr}	Lower threshold in response time
RT^{uprThr}	Upper threshold of response time
S^{rt}	Response time as per SLA
$SLAV$	Total SLA violation in an hour
$\triangle s$	Scaling Interval
sw	sliding window of size k
T	Throughput in last minute
U^{lowThr}	Lower threshold in CPU utilization
U^{uprThr}	Upper threshold of CPU utilization
VM^{AL}	Total virtual machines VM^P and VM^S
VM^F	Virtual machines required in future as per $\hat{w}_{t+1}$
VM^{max}	maximum on-demand VMs scaling limit
VM^P	Pending virtual machines list
VM^S	In service virtual machines list
w_t	Requests received at time t
$\hat{w}_{t+1}$	Analyzed future incoming request at time t
$\overline{w}_t$	Request answered at time t

4.4.1 Auto-scaling System Architecture

As per the proposed approach, the web applications in cloud architecture is shown in Figure 4.2. It is a communication architecture of the Application Provider (AP), Cloud Provider (CP) and end-user communication architecture. The end users send the request to AP through the Internet for Web applications. The load balancer has received the demand from AP and sends the request to the Virtual Machine (VM) deployed for

application-tier. The multi-tier web applications usually have three layers and a separate VM for each level. The user request may access all tier of web applications. Afterward, the user got the response this life cycle the user gets the answer. The auto-scaling method calculates the necessary VMs for the incoming requests. This research contributes to the auto-scaling mechanism in the analysis, planning and execution phase. In this model, the cloud provider considered which offers the infrastructure with different price policy, for example. On-demand and reserved.

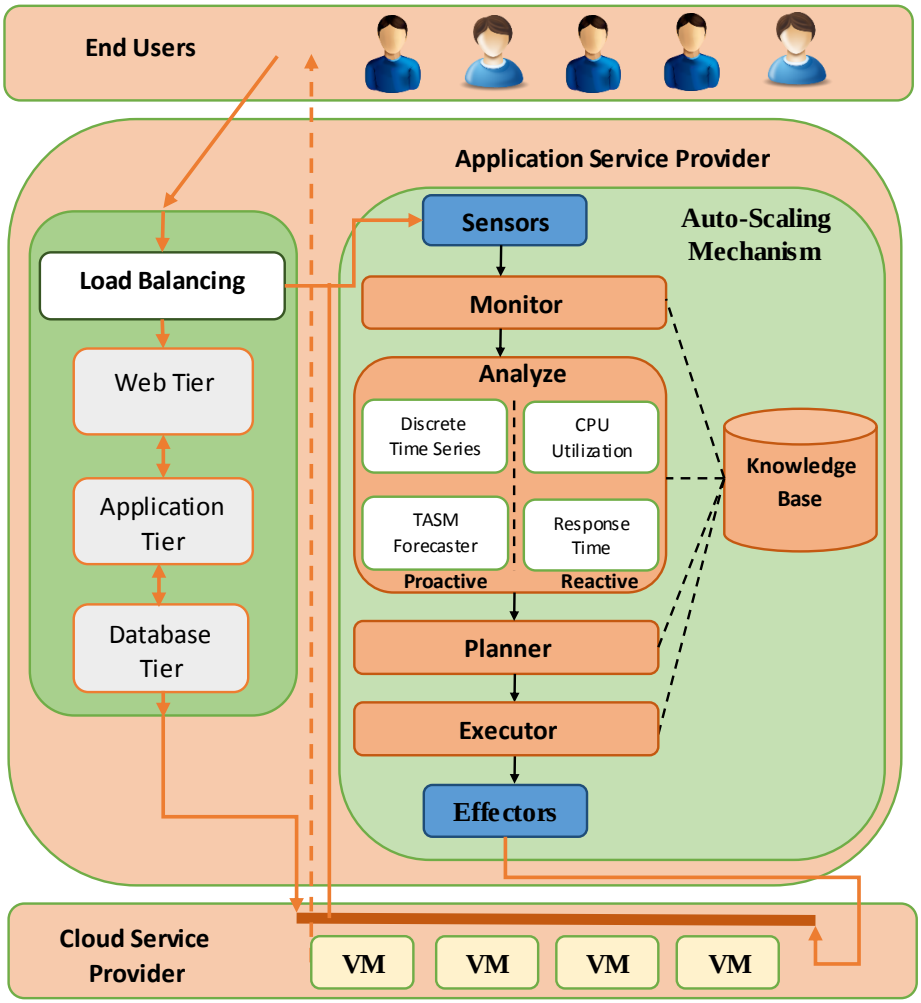

Figure 4.2: The cloud architecture for web applications

Algorithm 2 is defined as the mechanism used in our approach for auto-scaling. The first phase of this approach is to monitor the resources. This approach is unique as the decisions taken on scaling are reviewed in $\triangle s$ minutes called scaling interval. Every minute monitoring is done. The researcher mentioned the monitoring steps in Algorithm 3.

Algorithm 2 The pseudo code of auto-scaling management

```
Begin                                          ▷ Boot the reserved VMs for incoming workload
while system is running do
    for every 1 min do
        Monitoring();                                          ▷ Stores history of metrics
        if Clock % △s = 0 then
            Analysis(historical w of k size, history of M^u, history of M^rt)
            Planning(ŵ_{t+1}, A^u, A^rt);
            Execution(D);
        end if
    end for
end while
End
```

4.4.2 Monitoring Phase

The dynamic behavior of incoming workloads like seasonality, non-seasonality or flash crowds influences auto-scaling decisions. In Algorithm 3, the monitoring phase regularly fetches the information of application and infrastructure level parameters on a fixed interval [49]. After every minute, the monitoring took place. The parameters of application levels are end-user request (w), whereas parameters for the infrastructure level are the number of VMs and their usage. The response time is employed as an SLA parameter. The monitoring module logged the arrival rate, the available capacity and the capacity used with the control domain.

The Eq. 4.1 used to evaluate the response time of every request. In addition, Eq. 4.2 calculated the average response time. Here, the $ArrivalTime$ is the $Clock$ time when cloudlet is submitted and the $FinishTime$ is the $Clock$ time of the cloudlet completion. The $ProcessedTime$ is the total time for the cloudlet processing.

$$ResponseTime = FinishTime - ProcessedTime - ArrivalTime \tag{4.1}$$

$$M_t^{rt} = \frac{\sum_{j=1}^{TotalCloudlets^{Finished}} ResponseTime_j}{TotalCloudlets^{Finished}} \tag{4.2}$$

The Eq. 4.3 is used to calculate the average CPU utilization. Here, VM^{AL} is the addition of pending virtual machines (VM^{P}) and in-service virtual machine (VM^{S}).

$$M_t^u = \frac{\sum_{j=1}^{VM^{AL}} VM_j Utilization}{VM^{AL}} \tag{4.3}$$

Algorithm 3 The pseudo code of monitoring phase

1: /* User behavior parameters */
2: Store w_t ▷ Incoming workload (1 minute)
3: /* Infrastructure and platform level parameters */
4: Store VM^P, VM^S ▷ VM parameters
5: Store M_t^{rt} and M_t^u ▷ SLA parameters and Resource Utilization

4.4.3 Analysis Phase

In the chapter 3, the researcher described the proposed proactive analyzer to forecast a time series known as Technocrat ARIMA and SVR Model (TASM). This approach used the pattern discrimination technique on the sliding window of past workload [173]. In this research, a combination of reactive and proactive analytical techniques is presented to a new hybrid method of analysis. According to the Algorithm 4, the TASM employed (line no. 4) in a proactive section to forecast the maximum incoming request in a minute for upcoming scaling interval. In addition, the CPU usage and response time are analyzed for the reactive section (line no. 6).

Firstly, the time-series model are used to calculate the future arrival rate $\hat{w}_{t+1}$. The monitoring phase employed to stores the series of (w_t) in the $(\triangle s)$ time interval. The $w_s = \left(w_t^1, w_t^2, ..., w_t^k\right)$ is the time series represented in discrete model. TASM prediction model (line no. 5) calculated the $\hat{w}_{i+1}$, the flowchart of approach is shown in Figure 4.3.

Secondly, the average CPU usage of every minute obtained for each scaling interval (lines 7 to 9). The usage of the CPU is calculated according to Eq. 4.3. Moreover, the average CPU use assessed in line 10 as per Eq. 4.4 in the past scaling interval.

Thirdly, the last minute response time obtained from the monitoring phase. The response time is considered as the SLA parameter. The response time is evaluated according to Eq. 4.1. Then, the average response time of last minute with Eq. A_t^{rt} is calculated (line no. 11), it is a QoS indicator.

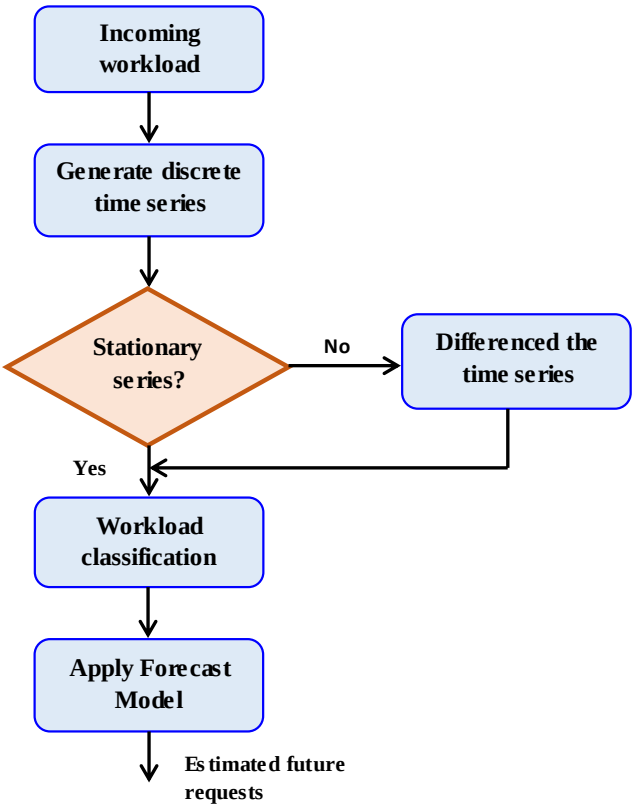

Figure 4.3: The workload forecasting approach using TASM prediction model

Thirdly, the response time is collected as an SLA parameter during the last minute collected from the monitoring phase. The response time is calculated as per Eq. 4.1. Afterward, the average response time calculates with Eq. 4.2. The average response time is a QoS indicator. In this approach, the researcher has considered the analyzed response time A_t^{rt} is the average response time in the last minute (line no. 11). The researcher thus placed the most emphasis on last-minute response time.

$$A_t^u = \frac{\sum_{t=1}^{k} M_t^u}{k} \tag{4.4}$$

4.4.4 Planning Phase

The planning phase makes decisions with auto-scaling techniques that are reactive and proactive. The reactive approach supported the flash load. The reliability of predictive testing remains in question since the workload does not depend on the historical workload all the time. This approach adds resources to the resource pool if available resources do not meet the adequate requirements. This approach ensures the capacity

Algorithm 4 The pseudo code of hybrid analysis approach (Proposed)

1: Input: M^u (duration is the last $\triangle s$ minutes), M^{rt} (duration is the last minute), sw (requests per minute from the past $\triangle k$ minutes)
2: Output: $\hat{w}_{t+1}$, A^u and A^{rt}
3: Variable: double $cpuUtilization \leftarrow 0$, string $predictionModel \leftarrow TASM$
4: /* Proactive Section */
5: $\hat{w}_{t+1} \leftarrow$ WorkloadPredictor($predictionModel$, sw) ▷ Predict the future arrival rate
6: /* Reactive Section */
7: **for** $i = 0 \quad to \quad i < \triangle s$ **do**
8: $\quad cpuUtilization \leftarrow cpuUtilization + M^u_{t-i-\triangle s}$
9: **end for**
10: $A^u \leftarrow cpuUtilization/\triangle s$ ▷ Average CPU utilization in $\triangle s$ minutes
11: $A^{rt} \leftarrow M^{rt}$ ▷ Response time of last minute
12: return $\hat{w}_{t+1}$, A^u and A^{rt}

available is higher than the capacity required.

The response time (A^{rt}) analyzed and CPU utilization (A^u) analyzed are evaluated under the threshold rules as per Algorithm 5. If the CPU usage analyzed is greater than the CPU usage of the top threshold value, and the response time analyzed is larger than the upper threshold value (line 4), an immediate decision is taken for $ScaleUp$ (line 5). Here, there is no additional consideration took place and return from the procedure (line 6). If the CPU utilization analyzed is less than the lower threshold value and the response time analyzed is lower than the response time threshold value then (line no. 7), the decision set on for *ScaleDown* (line no. 8) is a temporary one. This decision was filtered further through the proactive section (line no. 10) and then the final decision was taken.

$$VM^F = \left\lceil \frac{\hat{w}_{t+1}}{RT^{SLA}\mu} \right\rceil \tag{4.5}$$

The queuing model applied in proactive section (line no. 10). The number of VMs required for upcoming workload estimated using Eq. 4.5 (line no. 11). Here, three arguments are processing rate (μ), future incoming request ($\hat{w}_{t+1}$) and response

time (RT^{SLA}) according to SLA. Afterward, the current capacity compared with the calculated capacity required in the next scaling interval. If the estimated number of VMs is greater than the current capacity than decision set to $ScaleUp$ (line no. 13). Furthermore, if the calculated VMs are the same as available VMs than decision set to *DoNothing* (line no. 15), otherwise, the reactive auto-scaling decision is final to *ScaleDown* the VM.

The short-term workload prediction is applied to the pattern with less forecasting errors, whether the reactive approach used to handle the patterns with higher forecasting errors. The VMs are ready in a proactive approach before the incoming of actual workload and, the reactive approach takes the scaling decision as per the current status of the infrastructure level parameters. This proposed approach is able to correct the wrong of a proactive approach with reactive method.

Algorithm 5 The pseudo code of hybrid planning approach (Proposed)

1: Input: A^u, A^{rt}, U^{lowThr}, U^{upThr}, RT^{lowThr}, RT^{upThr}, $\hat{w}_{t+1}$
2: Output D
3: /* Reactive Scaling */
4: **if** $A^u > U^{upThr}$ and $A^{rt} > RT^{upThr}$ **then**
5: $D \leftarrow ScaleUp$
6: return D
7: **else if** $A^u < U^{lowThr}$ and $A^{rt} < RT^{lowThr}$ **then**
8: $D \leftarrow ScaleDown$
9: **end if**
10: /* Proactive Scaling */
11: $VM^F \leftarrow QueuingModel(\hat{w}_{t+1}, \mu, RT^{SLA})$ ▷ Queuing model $M/M/m$ estimates the future capacity
12: **if** $VM^{AL} < VM^F$ **then**
13: $D \leftarrow ScaleUp$
14: **else if** $VM^{AL} == VM^F$ **then**
15: $D \leftarrow DoNothing$
16: **end if**
17: return D

4.4.5 Execution Phase

The execution phase implements the perception of the planning phase. The final decision to scale-up, scale-down or do nothing is taken with the collaboration of cloud providers. The default executor randomly selects the machines from the resource pool to scale up/down. This phase cross-validates the VMs on-demand limit before the scale-up decision, which rejects the scale-up decision if it exceeds the limit. Likewise, no other request will be received if the on-demand resources limit is met.

4.5 Experiment Evaluation

The aim of this work is to devise the robust automated approach for scaling the resources with maximized utilization of the resources and provided response time according to SLA. The experiment was conducted as per the following:

- The comparison of prediction models AR, LR, MA, ARMA, ARIMA, SVR and TASM made for short-term prediction (1 minute). The actual workload traces are used to emulate the incoming requests. The $R-tool$ is used to implement the prediction models.
- CloudSim toolkit has extended to implement the existing and proposed techniques with new classes for auto-scaling features. The decisions of auto-scaling assessed on the basis of various performance metrics such as CPU utilization, response time, renting cost, SLA penalty cost and VM allocation.

4.5.1 Experiment Setup

The CloudSim toolkit extended with new classes and *R* scripts. The *R* tool is used to implement the prediction model. $R-Caller$ library used in CloudSim toolkit to integrate *R* API in Java. The different auto-scaling approaches are implemented such as threshold rules based technique [174], SVR [175], AR [30], ARIMA [122] and the investigator's proposed model TASM [173]. These prediction models compared on the basis of response time, CPU utilization, VM allocation and cost. The designed approach gave minimum response time and optimal resource utilization.

Table 4.2 describes the detail of weblog traces used in the experiment in this research. The experiment evaluation is performed with the following entities: cloud service provider, application service provider and end-user. The detail of the entities are:

Cloud Provider

The CloudSim offers the cloud provider ability for infrastructure related parameters and methods. Classes for rental management of resources are added in existing CloudSim toolkit. The time shared scheduling technique is used for the evaluation of proposed and existing models.

Application Provider

Cloud infrastructure is a platform for application providers to host the application. The on-demand resources in virtual machines are significantly delayed to start-up. This interesting study shows that VM start-up time varies with factors like VM request time, VM size and weekday. In literation, some authors are considered a normal or a fixed number distribution in VM start-up time. In this study, the researcher defined the VM startup time for the experiment is fixed at 5 minutes. Consequently, after every 10 minute, the hybrid auto-scaler made scaling decisions with the highest priority to reactive scaling. Although, the proposed technique is flexible and the scaling interval could be changed as per the need of user and application.

End User

The ClarkNet and NASA workload emulated as web application incoming request to AP. These weblog are incorporated in tremendous research studies for the evaluations of auto-scaling techniques [176, 177, 123, 132]. Table 4.2 describes the detail of weblog traces.

4.5.2 Results and Discussion

The 6 metrics for performance assessment are described as follows:

Table 4.2: Summary of datasets information

Dataset Name	**ClarkNet**	**NASA**
File name	Aug28log, access1	accesslogAug95
File size	171.0 MB, 172.5 MB	167.8 MB
Length (s)	3,328,587(14days)	3,461,612 (28 days)
Timestamp resolution	1 second	1 second
Sampling interval	1 minutes	1 minutes
Number of Samples	2880 (2 days)	2880 (2 days)

Prediction Accuracy of Time Series Models

The parameters of the cloud infrastructure are response time, CPU utilization and arrival rate of incoming requests are stored by monitoring phase. In the analysis phase, the key component is workload predictor. This phase forecast the future requests using the TASM prediction method based on the classification of discrete series in the sliding window [173]. This sliding window captures the latest historical request and prediction models are applied on the basis of the classification approach. In order to capture the various trends in incoming workload, the linear and non-linear models are employed. The model proposed shows a lower difference in incoming workload and predicted workload, which could be feasible for scaling decision for web applications requests in a cloud environment. The researcher tested the 10 minutes discrete series of predictions in previous work. In this experiment, the researchers tested the discreet set of 1 minute as shown in Figure 4.4 for the series ClarkNet and Figure 4.5 for the NASA series. The largest value of 1 minutes is picked in next scaling interval (e.g. 10 minutes) from the predicted workload. This strategy helps to reduce the SLA breach and optimize the appropriate resource usage in upcoming scaling interval.

Mean-Absolute-Percentage-Error (MAPE) and Root-Mean-Square-Error (RMSE) metrics are used to asses the accuracy of the prediction model. The Eq. 4.6 and Eq. 4.7 defines the standard metric for RMSE and MAPE. The obtained result described in Table 4.3 for ClarkNet series and Table 4.4 for NASA series. The experiment result showed that the proposed model gave good result in short-term (10 minutes) workload prediction. However, the short-term prediction (1 minute) is still a challenge for the prediction models. Thus, the prediction error could lead to SLA violation and delayed

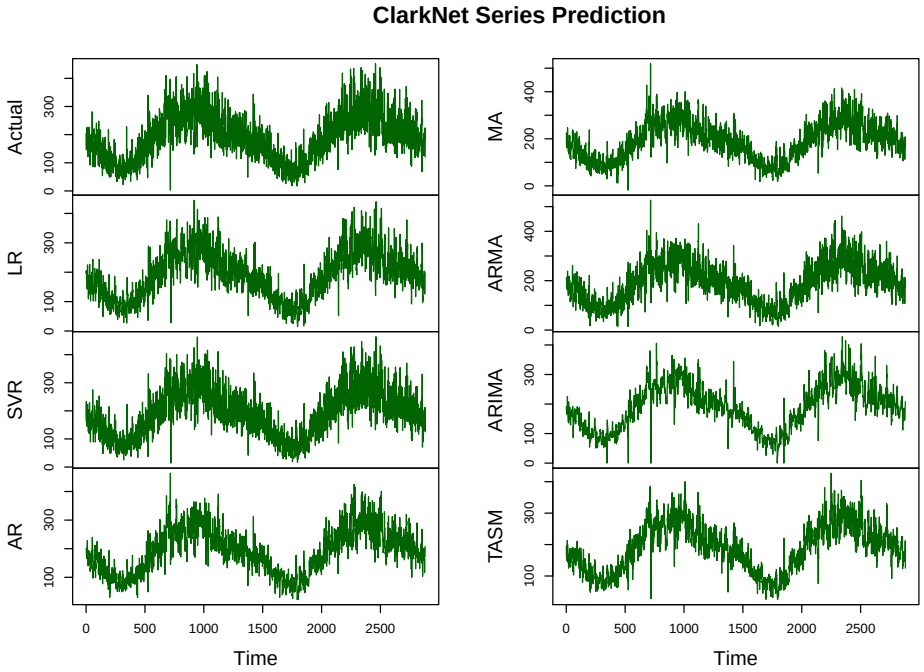

Figure 4.4: ClarkNet workload prediction using LR, SVR, AR, MA, ARMA, ARIMA and TASM

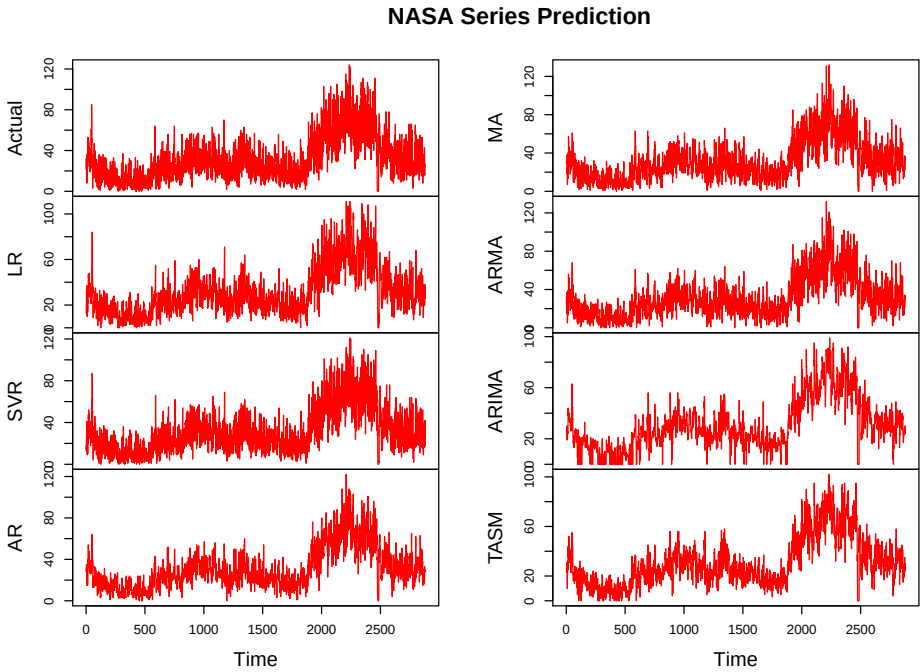

Figure 4.5: NASA workload prediction using LR, SVR, AR, MA, ARMA, ARIMA and TASM

Table 4.3: Accuracy of prediction models for ClarkNet workload.

Prediction model	**RMSE**	**MAPE**
LR	64.37	34.52
SVR	64.67	32.95
AR	54.60	30.42
MA	59.24	33.14
ARMA	59.25	32.98
ARIMA	51.24	26.64
TASM	42.24	20.63

Table 4.4: Accuracy of prediction models for NASA workload.

Prediction model	**RMSE**	**MAPE**
LR	16.56	66.77
SVR	17.0	67.93
AR	14.09	56.32
MA	15.72	62.40
ARMA	15.23	60.37
ARIMA	13.37	52.68
TASM	11.01	39.21

services. The researcher sensed the requirement of hybrid auto-scaling technique with a reactive approach. The proposed hybrid solution with the proactive and reactive technique could overcome the error of prediction models in scaling decision by taking the scaling action based on current resources utilization.

$$RMSE = \sqrt{\frac{1}{n}\sum_{s=1}^{n}\left(actualWorkload_s - predictedWorkload_s\right)^2} \tag{4.6}$$

$$MAPE = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{actualWorkload_s - predictedWorkload_s}{actualWorkload_s}\right| \times 100\% \tag{4.7}$$

VM Allocation

The number of VMs used to answer the web application request under hybrid analysis and planning algorithms are analyzed. The experiment conducted on the first couple

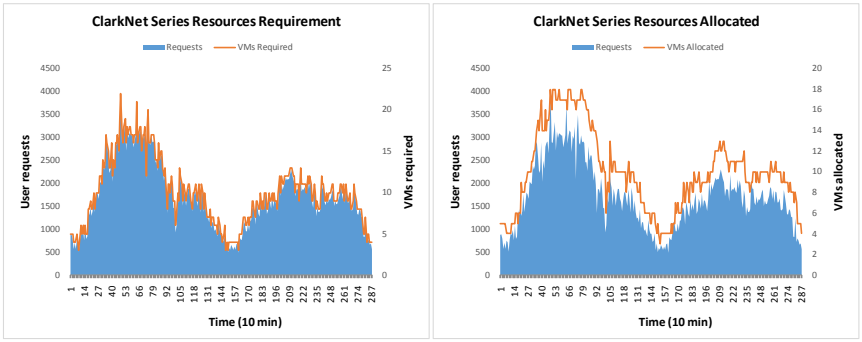

Figure 4.6: ClarkNet series VM required and allocated using proactive scaling

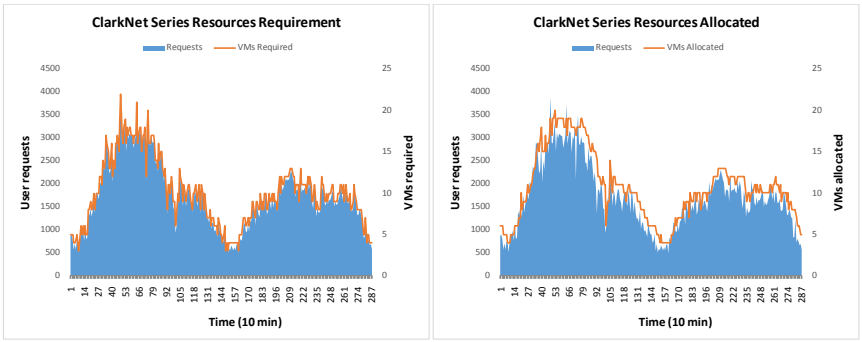

Figure 4.7: ClarkNet series VM required and allocated using proposed RHAS

of days of ClarkNet and NASA discrete time series. The performance of the analysis phase showed that the performance of the proposed prediction model TASM is good as compare to the state-of-the-art prediction model for web applications. The proposed hybrid planning algorithm tested with the help of two scenarios. The comparison result shown in Figure 4.6 is for the first experiment with proactive auto-scaling with TASM and Figure 4.7 with the proposed hybrid solution with ClarkNet series. Similarly, Figure 4.8 and Figure 4.9 show the second experiment carried out with proactive TASM and the proposed auto-scaling approach on NASA series. The TASM model is capable of predicting future demand in peak hours. However, due to predictive error, resource oscillation is present in a proactive approach. The AP rents VMs on request and releases with workload reduction, frequent releasing and acquiring of VM depict the poor scaling approach and lead to delayed services and more cost in the hourly billing cycle. The proposed approach shows the stability in the scaling process reduce the scaling overhead and proper utilization of resources under various billing cycles. This also helps to reduce the renting and SLA penalty cost.

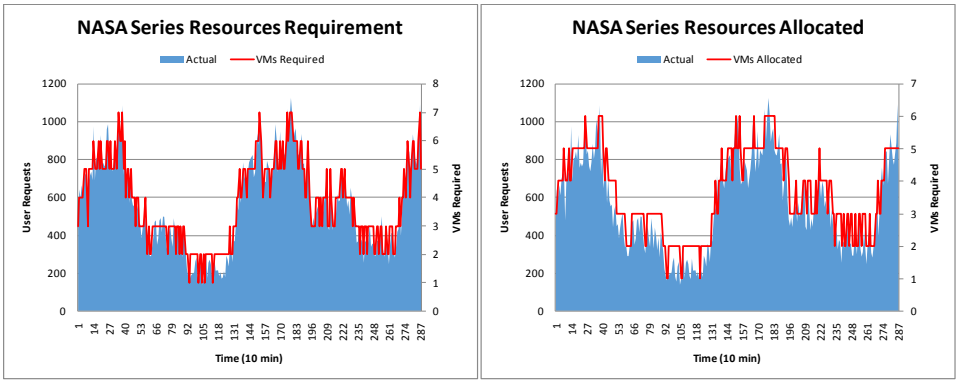

Figure 4.8: NASA Series VM required and allocated using proactive scaling

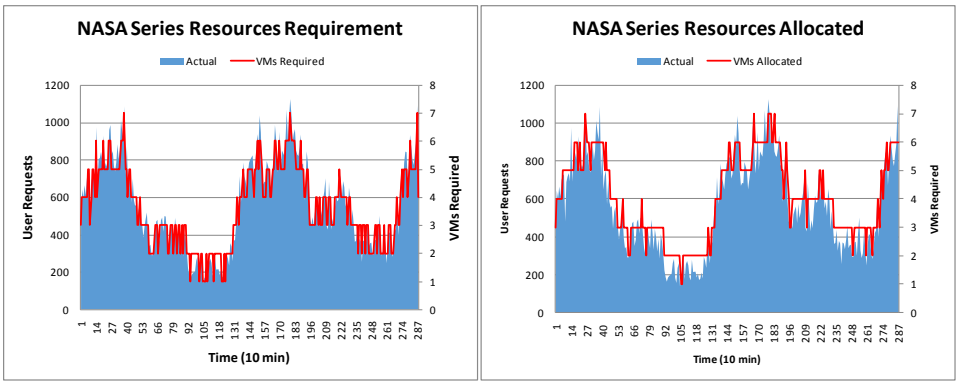

Figure 4.9: NASA series VM required and allocated using proposed RHAS

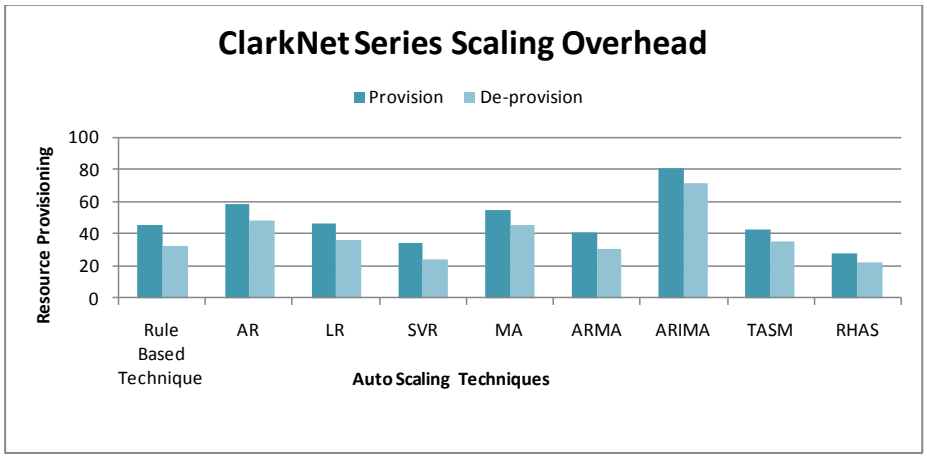

Figure 4.10: ClarkNet series scaling overhead

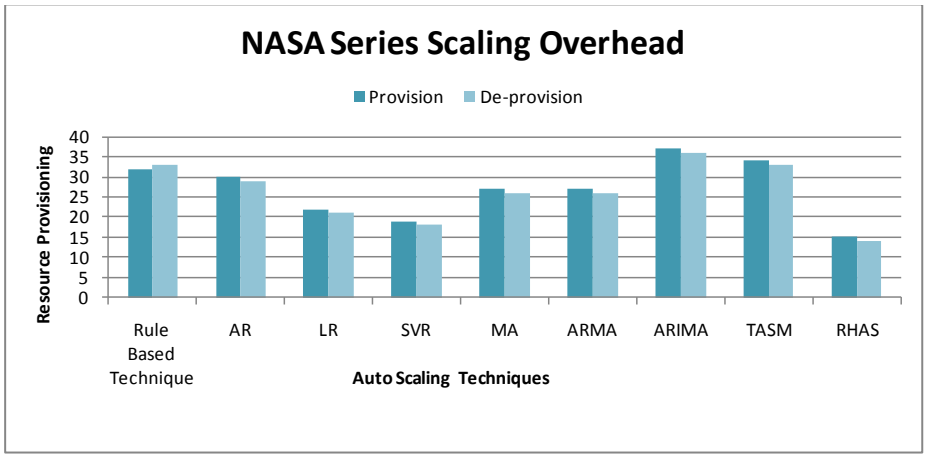

Figure 4.11: NASA series scaling overhead

The ClarkNet and NASA series again used for the third and fourth experiment. The $6th$ and $7th$ day in the series used to evaluate the performance of proposed algorithms. Figure 4.10 and Figure 4.11 shows the experiment results for the number of the scaling decision took under various prediction approaches. Furthermore, the proposed planning algorithm reduces over-usage and under-usage of resources up to 16 percent.

CPU Utilization

The CPU utilization means work is done by the Central Processing Unit (CPU) for the VM deployed in the cloud environment. The proposed mechanism RHAS has been tested for their performance with the percentage of CPU utilization while executing the incoming workload. The use of the CPU was tested for the prediction models, threshold rules based technique and proposed RHAS technique. The results of the experiment for ClarkNet series is shown in Figure 4.12 and Figure 4.13 for shows comparison result of NASA series. The technique based on the threshold rules is capable of achieving maximum use of the CPU in the ClarkNet web trace, but for NASA series due to the 10 minutes scaling period, the Threshold rules (TR) mechanism CPU utilization achieved only 42 percent. This technique is also faces lot of under-provisioning and over-provisioning issues, due to which cloud provider is satisfied but the application provider observed often downtime in application availability. The proposed technology is a mixed reactive and proactive approach, thus providing a consistent usage of 90 percent of the CPU in both ClarkNet and NASA series.

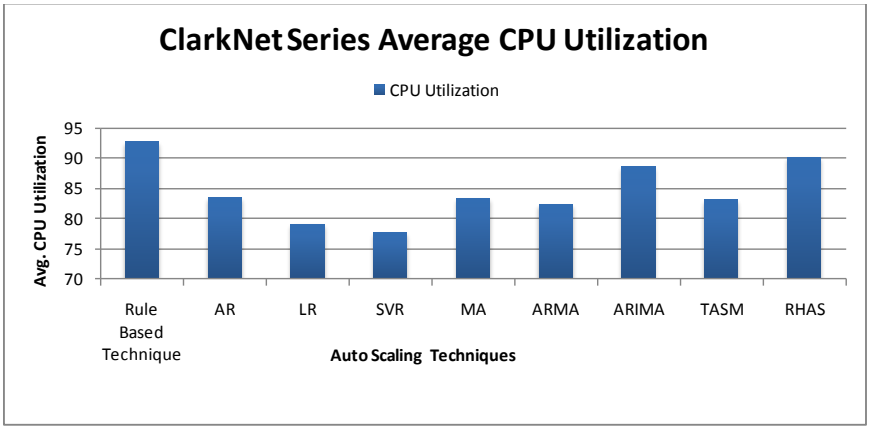

Figure 4.12: ClarkNet series average CPU utilization

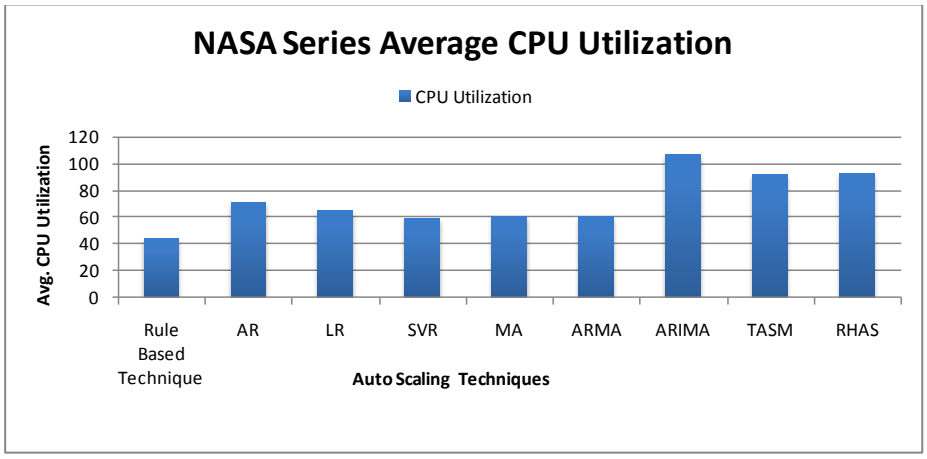

Figure 4.13: NASA series average CPU utilization

Response time

The elapsed time between the application request to response is known as response time. The web application response time should be fast. In this research work, the requested response time in this experiment was taken 1 *second* according to the SLA. One of the Quality of Experience (QoE) parameter is the minimum response time. The ClarkNet series result is shown in 4.14 and NASA series result described in Figure 4.15 are the representing the comparison of the outcome of the mean response time. The proposed algorithms in this research are given a response in less time span. The proposed technique RHAS is able to provide better QoS to the end-users under static and dynamic workload conditions.

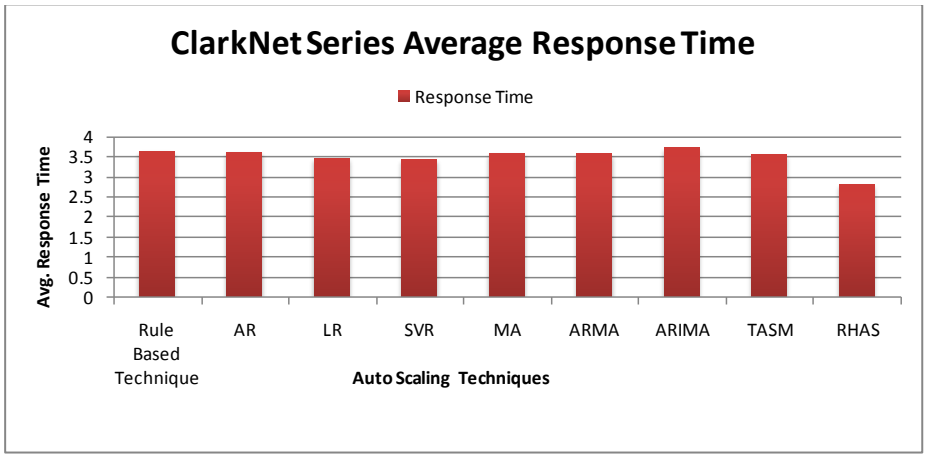

Figure 4.14: ClarkNet series average response time

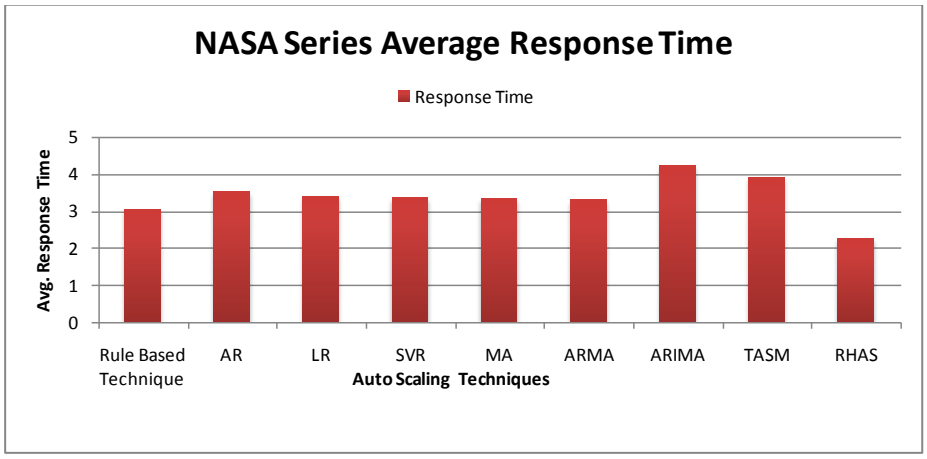

Figure 4.15: NASA series average response time

SLA violation

The SLA agreement is a crucial aspect that ensures cloud services are provided to end-users. In the event of an SLA violation, the AP must pay a penalty to the end-users. The violation of the SLA calculated with Eq. 4.8.

$$SLAV = \sum_{i=1}^{w} RT_i - S^{rt} \tag{4.8}$$

The comparison result fo ClarkNet series shown in Figures 4.16 and NASA series shown in Figure 4.17 against SLA violation. The results of the experiments show that the proposed RHAS technique have less than 1% SLA violation. Thus, the proposed

mechanism can give better QoS to the end-users as compare to state-of-the-art auto-scaling techniques for web applications.

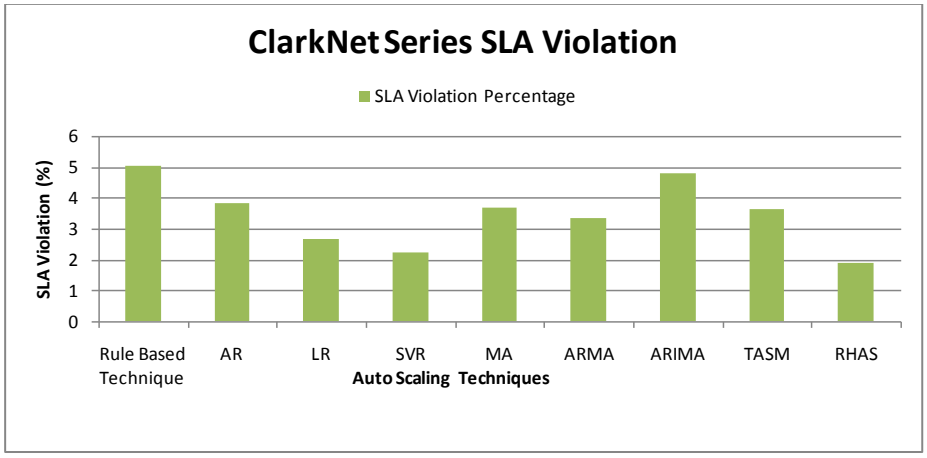

Figure 4.16: ClarkNet series SLA violation

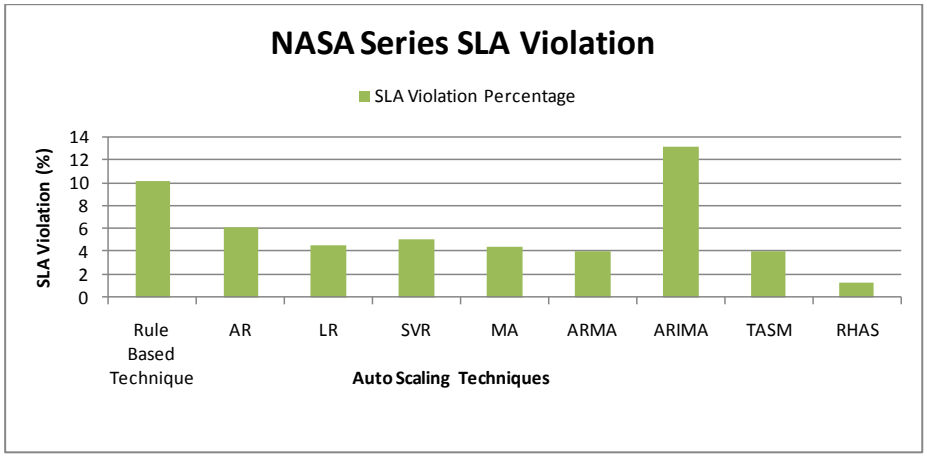

Figure 4.17: NASA series SLA violation

Renting Cost

The resources rented on-demand as per the pay-as-you-go model. The rental cost is calculated hourly basis by Amazon EC2. The total amount is the sum of each VM per hour usage. Figure 4.18 are shown for the rental cost of the ClarkNet series and Figure 4.19 is shown the renting cost of the NASA series. This infringement causes the application provider to pay a penalty. The proposed model in this research is cost-effective for application providers. The combined renting and penalty cost reduces up

to 26 percent.

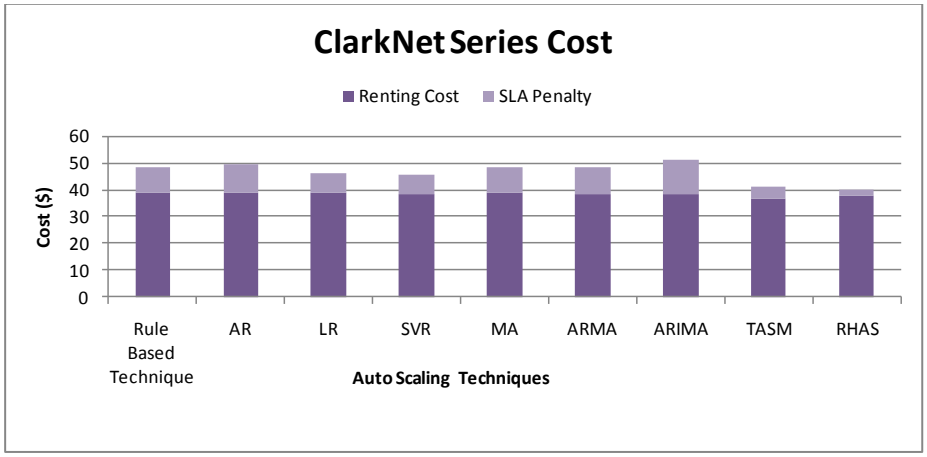

Figure 4.18: ClarkNet series overall cost

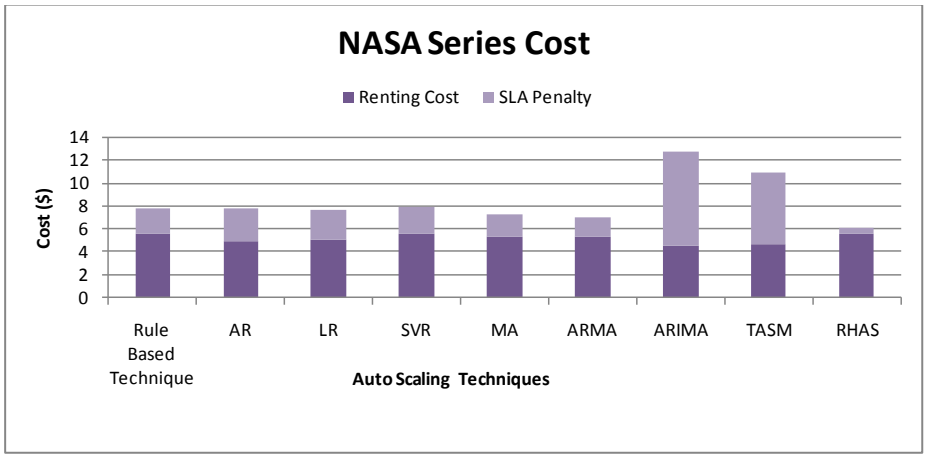

Figure 4.19: NASA series overall cost

It is worth to mention that the proposed technique RHAS for auto-scaling is a robust approach, providing the end user with an efficient and fair amount of QoS while benefiting ASP. Reduction of the SLA penalty will minimize the total bearing cost of ASP. The end-user will experience a justified response time QoE according to the SLA agreement.

4.6 Summary

The web application providers experienced irregular load changes in the cloud environment. This issue leads to an unsure scaling decision. In this study, the researcher

designed a hybrid analysis and planning algorithms under Robust Hybrid Auto-Scaler (RHAS) approach. The proposed RHAS technique provides benefit for auto-scaling with important terms such as costs and QoS parameters. The results from the experiments showed that it reduced the rental cost and SLA violation as compare to state-of-the-art proactive and reactive scaling methods. There is also a fair quantity of CPU usage in the proposed technique. Consequently, it was evident that other parameters like the number of requests and the use of CPU are equally important in the scaling decisions along with response time. The RHAS approach can offer the application providers profit in terms of cost and also gives the end user QoE for web application in the cloud environment.

CHAPTER 5

A PROFIT-AWARE RESOURCE PROVISIONING FOR WEB APPLICATIONS IN CLOUD

5.1 Introduction

The paradigms for cloud computing include Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [11]. In a cloud environment, VMs are provided with different pricing models such as the reserved, spot instances and on-demand [178]. The Application Service Providers (ASP) are rent out the VMs from Cloud Service Providers (CSP) and host the applications. These applications are SaaS applications such as e-commerce applications. The customers use Internet services to access the web application from cloud data centers [179]. The requests of end users are dynamic and expect service quality. The careful design of the techniques to supply resources could overcome the problem of resource oscillation, this situation arises when the resources are over-utilized or under-utilized in the data centers [180].

The exact number of resources estimation is an important decision for web application in the cloud. This factor is influenced by the user request pattern [181]. The irregular access pattern of the websites generates the fluctuation in the incoming traffic.

The number of resources if deployed in the cloud data center for incoming traffic, this leads to the over-utilization, due to this ASP has to pay more cost. The second condition, when less number of resources arranged for the incoming traffic. This lead to an under-utilization state, where renting cost is less but ASP has to pay the SLA violation penalty to the customers [5]. Thus, a need arose to develop a technique which can overcome both the state and give a fair amount of services to the end-users.

The resource oscillation problem could be solved with dynamic provisioning technique for the data center resources. This technique can be designed with a combination of infrastructure-level parameters and user-level parameters. The ASP requirements are maximum profit gain with minimum SLA penalty. The Quality of Service (QoS) parameters help to strengthen the reliability of customers on the service providers.

In chapter 3, the researcher discussed the proposed Technocrat ARIMA and SVR Model (TASM) [182]. The proposed model highly suited for the applications with the dynamic workload. In this research work, the researcher combined the technique developed in chapter 3 and chapter 4 to develop the Triangulation Resource Provisioning (TRP). The proposed technique in this research used the feature of robust auto-scaling technique with the enhanced execution module. In this study, the researcher contribution is as per the following:

- Designed a resource provisioning mechanism for web applications in a cloud environment.
- Designed and developed the profit-aware surplus VM selection technique.
- The real weblog traces are used to evaluate the proposed and existing techniques on various performance metrics such as CPU utilization, response time, VM allocation and cost.

5.2 Background

The autonomous computing is able to perform self-configuration, self-optimization, and self-protective [23, 183]. In the autonomous scaling, the researchers in history used the same technique and develop the provisioning approach with four phases of Monitor-Analyze-Plan-Execute (MAPE) loop [24]. The MAPE loop in cloud architecture is

shown in Figure 5.1. The infrastructure and user behavior parameters are collected by monitoring phase [25]. Afterward, the analyze phase considered this data to process with reactive and proactive scaling algorithms to find useful information. Furthermore, the third phase took this information to plan the scaling decision to scale-up, scale-down or do-nothing. The fourth phase is the execution phase, which implements the scaling decision as per the planning phase decision and providers constraints such as the limit of on-demand VMs to deploy for the particular application. In literature, auto-scaling techniques are categorized as threshold rules, control theory, queuing theory, application profiling, fuzzy logic, and time-series based [31].

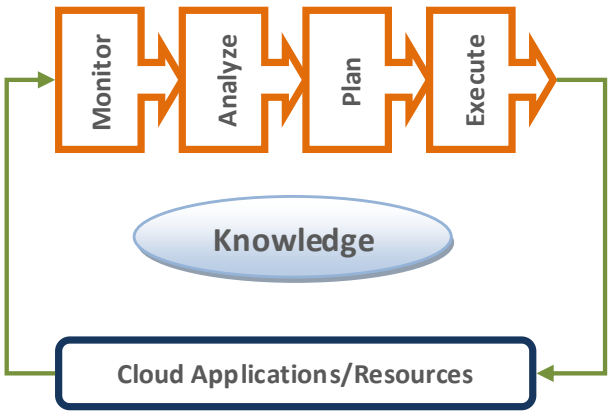

Figure 5.1: MAPE-K based cloud resources management.

5.2.1 Monitor

This phase collects the data from the cloud data centers about the resource utilization, Service Level Agreement (SLA) violation and users incoming requests [116, 184]. The cloud providers offer the monitor services on the hypervisor level, VM level, and application level.

5.2.2 Analyze

The data received from the monitoring phase processed to find useful information. This information helps in the decision-making process in the planning phase. The two types of analysis technique are reactive and proactive, which is used in literature by various authors. In the reactive approach, threshold values are decided for scale in/out decision

in horizontal scaling [185]. In a proactive technique, the incoming traffic or resource utilization is forecast from the historical data. This information used to percept the future situation of workload and resource usage. The proactive technique is able to dismiss the issue of VM delay in a startup, while the reactive approach is efficient to handle the current uncertain situation.

5.2.3 Plan

This phase collects the processed information from the analysis phase. Here, the policies are designed to take the scaling action based on the arguments such as the number of request in the next scaling interval, average CPU utilization, average response time, etc. In literature, numerous techniques are designed based on the QoS, SLA, cost, load, and resource to overcome the resource oscillation in cloud infrastructure [186, 187].

5.2.4 Execution

The cloud providers give unlimited resources flexibility to the application providers. According to the budget and deadline constraints, the ASP is deciding the limit of maximum on-demand resources. The execution phase provides the feature to pass the scale-in or scale-out decision according to the limit set by the ASP. Aslanpour et al. [132] designed the super-professional executor, which provide the feature of quarantined the spare capacity if the billing hour is not completed. This technique can save the cost in hourly billing cycle cloud services.

5.2.5 Knowledge

The shared repository is called a knowledge phase in the MAPE-K loop. This stores the data related to the status of resources, historical logs and policies [188]. The autonomic manager applied this information at various events in the decision-making process.

5.3 Related Work

The brief survey of resource provisioning techniques is discussed in this section.

A survey on the autonomic computing using MAPE-K conducted to introduce various techniques. In this study, Huebscher and Julie [189] introduced the basic concepts

and motivation for autonomous computer applications. Rahimizadeh et al. [190] devised a two-tier application with the MAPE approach in dynamic cloud provisioning environment. Maurer et al. [191] has developed the MAPE autonomous control circuit for the cloud infrastructure. The adaptive technology designed based on thresholds values. In the present study, the researcher used a hybrid approach with the prediction model and threshold rules for scaling actions. The scientific application's configuration performed inefficient manner with the MAPE loop using adaptive framework [192]. Ghobaei et al. [193] also applied the MAPE-K loop in provisioning of resources in cloud data centers.

Islam et al. [125] applied neural network and linear regression for the resource usage analysis. The author also discussed the issues in resource provisioning of cloud data centers. Huang et al. [116] used Double Exponential Smoothing (DES) to calculate the resource utilization with scaling indicators. Bankole et al. [152] and Ajila et al. [194] performed analysis using neural network. The performance metrics throughput, response time and resource usage are analyzed. Pankaj Deep et al. [186] develop the resource provisioning techniques for the analysis and planning phase. The effectiveness of indicators enhanced with the help of a decision tree. Mohamed et al. [185] developed a reactive approach based on the MAPE loop with reactive scaling policies. The static policies is a problem for dynamic resource provisioning, especially for web applications in cloud computing. Herbst et al. [195] proposed a proactive analysis phase along with monitoring phase in the MAPE loop. Simple to complex attributes applied to choose the prediction model with fewer errors. Singh et al. [196] designed resource allocation technique based on the clustering of workload. The author segregates the workload as per the QoS requirements. Toosi et al. [197] developed a threshold-rule based heuristics to use renewable energy for load balancing in the cloud.

Casalicchio and Silvestri [198] proposed the 5 rules to calculate the resources for various architectures. Garcia et al. [199] designed planning phase with rules based on SLA agreement. This approach aimed at improving the QoS at a minimal cost. A planning algorithm designed to analyze the capacity of the resource using machine learning approach [184]. Fallah et al. [74] used a reactive and proactive approach in the planning phase to estimate the allocation of resources with learning automata. Beltran et al. [187] devised approach with vertical and horizontal scaling combined. Further, an auto-

nomic framework for resource provisioning reduces the cost of application deployment. Molto et al. [200] applied over-subscription and live migration techniques for vertical and horizontal scaling. Arani et al. [201] introduced the MAPE loop based framework for resource provisioning. Aslanpour et al. [202] designed the proactive and reactive based auto-scaling approach. Two-sided planning phase devised with the resource utilization and SLA violation. Moldovan et al. [203] proposed the cost-aware policies for surplus VM selection in a cloud environment. The author described the importance of resource usage and CPU utilization for scale-out decisions.

In literature, many authors proposed the resource provisioning techniques to strengthen the SaaS. The frameworks, mechanism, models, and algorithms are developed which specially focused on a web application in the cloud environment. The key challenge in reactive auto-scaling approach is VM startup time, which varies from 5 minutes to 15 minutes [186, 204, 205]. The wisely applied prediction model with a reactive approach could reduce the scaling overhead and enhance the resource utilization.

In the present study, the researcher designed the architecture for resource provisioning for web applications. The proposed technique is a hybridization of proactive and reactive techniques.

5.4 Proposed Approach

The proposed approach for resource provisioning is discussed in this section. The approach developed in the present work is based on the IBM MAPE-K loop and the modules were executed at a specified interval. The triangulation approach employed a proactive and reactive resource scaling methods. The cost and SLA trade-off is considered in the proposed provisioning mechanism. The goal of the present study is to designed a profit-aware resource provisioning approach for ASP while giving the QoS to the end-users.

The proposed cloud resource provisioning architecture is shown in Figure 5.2. The ASP, CSP and end-users are the crucial components of the cloud environment.

5.4.1 Monitoring Phase

In the monitoring phase, the resource-aware approach collects the for resource usage (e.g. CPU utilization) [180] [23] and, SLA-aware approach monitored the response

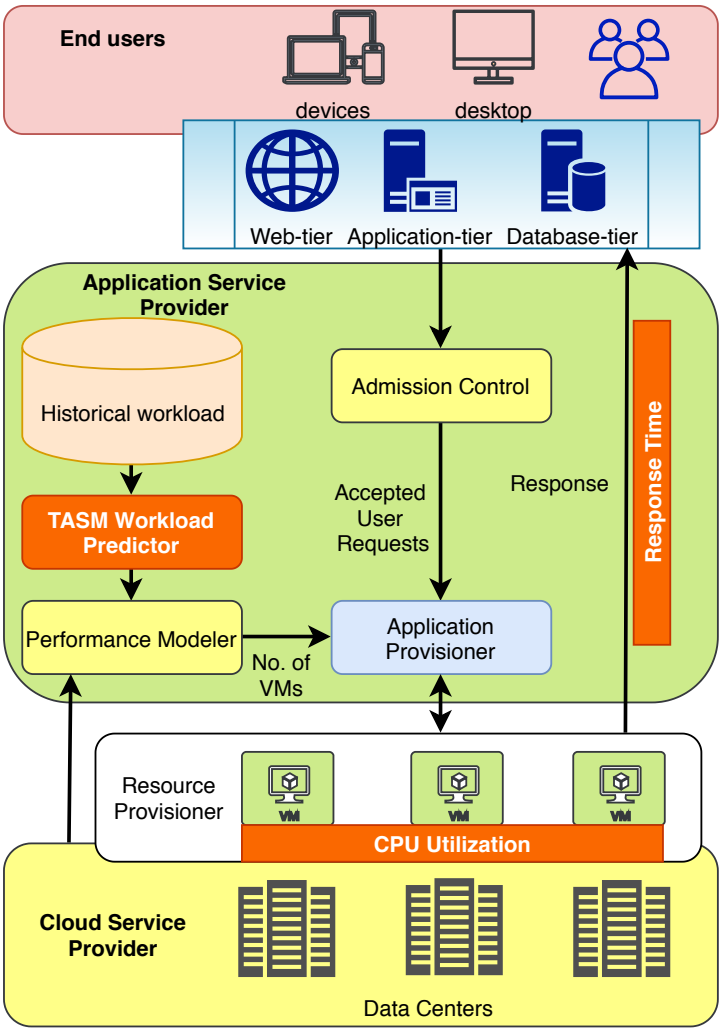

Figure 5.2: The cloud resource provisioning architecture.

time. The information related to incoming requests, SLA parameters and resource utilization helps to take the effective provisioning decisions. The cloud architecture shown in Figure 5.2 highlighted the monitoring parameters: workload prediction, response time and CPU utilization.

5.4.2 Analysis Phase

The data collected from the monitoring phase used in the analysis phase to find the useful inferences from the data. The historical workload, CPU utilization and response time are the key components of proposed Triangulation Resource Provisioning (TRP) mechanism. The performance modeler estimates the number of VMs as per the analysis discussed in Chapter 4 and Section 4.4.3. The proposed TRP mechanism shown in Figure 5.3. The working of each module is describe as follows:

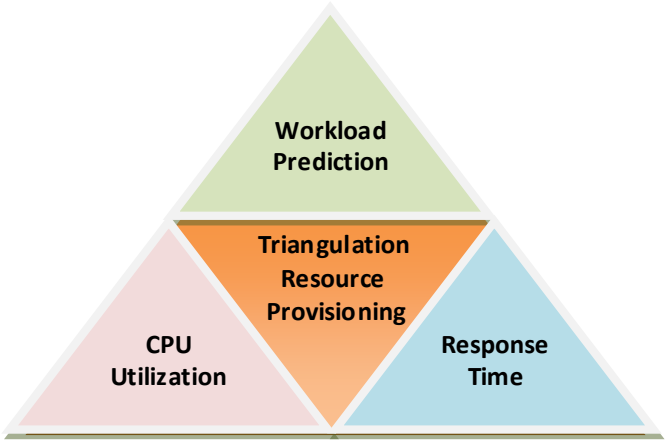

Figure 5.3: The proposed triangulation resource provisioning (TRP) approach.

Workload Prediction

In the chapter 3, the researcher described the working of proposed prediction model for web application in cloud named as Technocrat ARIMA and SVR Model (TASM) [182]. The prediction accuracy of the proposed TASM model is better as compare to the state-of-the-art prediction models. In the present study, TASM workload predictor employed in the workload predictor module in the cloud architecture. The workload prediction approach is shown in Figure 5.4. The repository stores the historical workload, the researcher collect the information and convert the incoming request in discrete time series. The detail working of the approach is described in Chapter 3 and the published article [182]. In addition, the discrete time series further classified in slow and fast scale. The proposed classification approach selected the ARIMA, LR or SVR model based the workload pattern. In the Chapter 3, the proposed model test on the discrete series of 10 minutes. In the current study, the researcher consider short term prediction of 1 minute discrete time series. The result of 1 minute prediction approach discussed in Chapter 4.

CPU Utilization

CPU utilization is the key parameter of resource usage. This can be observed from the host deployed the VMs in cloud architecture. The average CPU utilization is calculated according to Eq. 5.1.

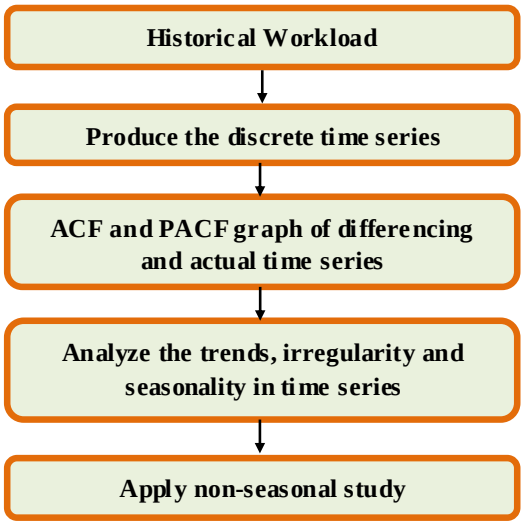

Figure 5.4: The TASM analytical process for workload forecasting.

$$AverageCpuUtilization = \frac{\sum_{i=1}^{OnlineVms} VM_i CPUutilization}{OnlineVMs} \tag{5.1}$$

Where VM_i utilization is the CPU utilization of VM_i and, $OnlineVMs$ are the total VMs including in-service VMs and startup VMs.

$$VM_i CPUutilization = \frac{\sum_{j=1}^{currentcloudlets} (CloudletPEs_j * CloudletLength_j}{PE * MIPS} \tag{5.2}$$

The $CurrentCloudlets$ are total incoming request in the scheduler according to discrete time series (e.g. the number of request in 1 minute). The $CloudletPEs$ are the total processors requirement and a number of instruction in each user request $cloudletLength$. The Million Instruction Per Second ($MIPS$) and Processing Elements (PE) defined the processing capacity of each VM in cloud data centers.

Response Time

The response time is the duration taken by the system to respond to each request in the cloud. The Eq. 5.3 used to calculate the average response time in the past minute in the current scaling interval. Response time is also an SLA parameter usually considered by the end-users. The surplus VM selection policy also used the response time for a scale-in decision.

$$AverageResponseTime = \frac{\sum_{i=1}^{TotalProcessedCloudlets} ResponseTime_i}{TotalProcessedCloudlets} \quad (5.3)$$

The number of cloudlets executed in last minutes represented with $TotalProcessedCloudlets$. $ResponseTime_i$ defined the delay in response iat i^{th} response to the incoming request. Furthermore, the Eq. 5.4 used to calculate the response delay.

$$ResponseTime_i = FinishTime_i - ProcessTime_i - ArrivalTime_i \quad (5.4)$$

The $ArrivalTime_i$ is the recorded $Clock$ when the user request accepted in a cloud system, the cloudlet processing time represented with $ProcessTime_i$. Once the cloudlet execution is complete then system $Clock$ time recorded is the $FinishTime_i$ of each cloudlet.

5.4.3 Planning Phase

The application provisioner implements the planning phase. The planning module was based on the threshold based rules to get a decision on scaling as the planning phase described in Chapter 4 and Section 4.4.4. This phase used the analyzed information during the analysis phase.

The threshold rules are designed to take scaling decision as per the scaling indicators. The processed parameters collected from the analysis phase such as predicted workload in scaling interval, average CPU utilization and response time. The response time is set $0.2s - 1.0s$ for lower and upper threshold and CPU utilization is set as $20\% - 80\%$ respectively. Rather, the proposed approach is flexible and application provider can change the values as per the QoS requirements.

The planning phase takes the decision for scale-in, scale-out or does nothing. The upper-threshold values are compared against the current CPU utilization and response time, if either of the analyzed parameters found greater, further forecast current workload compared with the forecast workload. The $scale-out$ decision occurred immediately, if the current $cloudlets$ are less than forecast $cloudlets$.

Otherwise, if CPU utilization and response time both are less against lower-threshold values then further compare the current number of requests with the predicted workload. The $scale-in$ decision taken in case the forecast requests are less than the current requests, else the decision would be to $do-nothing$.

This decision of $scale-in$, $scale-out$ and $do-nothing$ input to execution phase.

5.4.4 Execution Phase

The resource provisioner implements this phase and took the scaling decisions such as scale-in, scale-out or do nothing as per the decision of the planning module. The present study aims to optimize the scale-in decision in a cloud environment.

The cloud environment has the elasticity feature, due to which de-provisioning occurred in the scale-in decision. The present study presented a profit-aware policy for scale-in decision in a cloud computing environment. The approach is flexible for providing the scale-in decision for any type of billing cycle. The experiments in this research work done on the Amazon EC2 hourly billing services. The detailed working of the proposed surplus VM selection defined in Algorithm 6. Table 5.1 described the symbols used in this section.

As per proposed algorithm 6, two important factors are considered, first one is pending minutes from the billing cycle from the VM and second is the number of request in the queue to be processed on the VM also called load on the VM. The line no. $(4-11)$ are designed to create a list of VMs whose billing cycle will be terminated in the next scaling interval represented with $onDemandSurplusVMs$. In line no. $(12-13)$, the selection of VM is limit to specific VM only if the $onDemandSurplusVMs$ have only 1 VM. In this condition, the approach is designed to select the only VM present on the list. Furthermore in line no. $(14-15)$, the surplus VM list may be empty because no VM has near to the completion of billing cycle duration $\triangle s$, then no scale-in event would not occur. In addition, if the $onDemandSurplusVMs$ size is more than 1, the selection of VM out of the list. In line no. $(17-29)$, the eligible VM for scale-in further sorted based on the load on the VMs. To ensure the QoS to end-user, the selection of VM with least load is selected for scale-in event. The $calcuatePendingST$ function employed to estimate the cloudlets service time. In line no. (30), the $surplusVM$ find during the selection process returned to the resource provisioner.

5.5 Experiment Evaluation

CloudSim [97] toolkit is used to simulate the the proposed model TRP. Cloud infrastructure can be simulated with CloudSim. Data analytic task performed with libraries

Algorithm 6 The pseudo code of proposed profit-aware surplus VM selection policy

1: Input: OnDemandVM List $(VM^S + VM^P)$
2: Output: surplusVM
3: Variables: $availableTime, remainingTime$, $onDemandSurplusVMs \leftarrow null$, $minLST \leftarrow anHour$
4: **for** vm in onDemandVmList **do** ▷ Prepare the VMs list having left service time less than or equals to scaling interval
5: $LST \leftarrow null$;
6: $availableTime \leftarrow Clock - VM.getRequestTime()$
7: $LST \leftarrow anHour - (availabletime\%anHour)$
8: **if** LST$<= \triangle s$ **then**
9: onDemandSurplusVMs.add(vm);
10: **end if**
11: **end for**
12: **if** onDemandSurplusVMs.legnth() == 1 **then** ▷ If list containing only 1 VM
13: $surplusVM \leftarrow onDemandSurplusVMs.pop()$ ▷ Select the first VM as surplusVM
14: **else if** onDemandSurplusVMs.empty() **then** ▷ If list is empty
15: surplusVM $\leftarrow null$
16: **else** ▷ If list containing more than 1 VMs
17: **for** vm in onDemandSurplusVMs **do**
18: $vm.LST \leftarrow 0$
19: $cloudletList \leftarrow vm.getScheduler().getCloudletList()$
20: **for** cloudlet in cloudletList **do**
21: $pendingServiceTime \leftarrow calculatePendingST(cloudlet)$
22: $vm.LST \leftarrow vm.LST - pendingServiceTime$
23: **end for**
24: **if** $abs(vm.LST) < minLST$ **then** ▷ abs stands for absolute value
25: $minLST \leftarrow vm.LST$
26: surplusVM $\leftarrow$ vm
27: **end if**
28: **end for**
29: **end if**
30: return surplusVm;

Table 5.1: Description of notations

Parameter	Description
$Clock$	Timer for Simulation
$\triangle s$	Scaling Interval
OnDemandVM	Combined List of in-service and pending VMs
VM^P	List of VMs about to ready or in ready state
VM^S	List of VMs in-service
$OnDemandSurplusVMs$	List of potential surplus VMs in scaling interval
LST	VM's left service time in current billing hour
$cloudletList$	List of incoming request called cloudlets
SurplusVM	In Algorithm 6, the VM finally selected for scale down

Table 5.2: Detail of ClarkNet dataset

Parameter	Value
Log Files	Aug28log, access1
No. of Requests	3,328,587
Duration	2 weeks
Discrete Time series	1 minute

enriched R Tool. CloudSim toolkit extended with auto-scaling classes and data science libraries. *R* language scripts are executed in the CloudSim environment with *R* Caller library.

5.5.1 Experiment Setup

The experiment was conducted using the weblog of ClarkNet [163]. The series contains different workload patterns of the time series helped to evaluate the performance of existing and proposed provisioning mechanisms in normal and peak hours. The workload details presented in Table 5.2.

In cases of peak workload, larger VMs are provisioned, whereas small VMs are provisioned when workloads are normal. The amount of time to start VM considered as 5 minutes for this experiment. The scheduling algorithm is considered in the experiment is time shared.

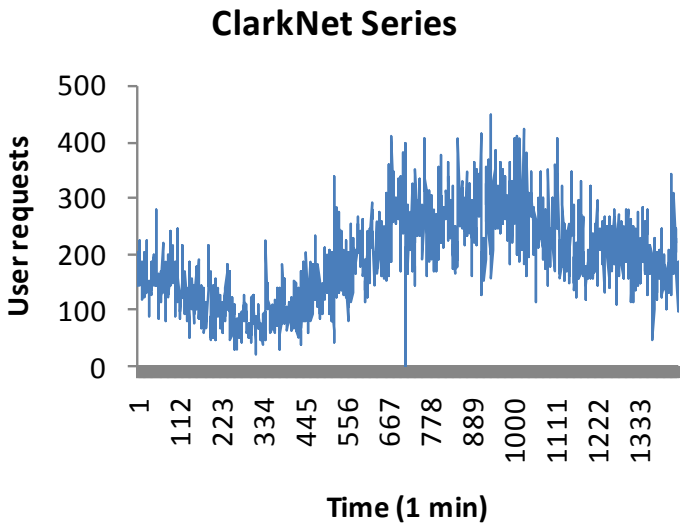

Figure 5.5: ClarkNet workload of the 6^{th} day in week.

5.5.2 Performance Evaluation

The performance metrics applied to evaluate the proposed resource provisioning mechanism are as follows:

Response Time: The time to respond to the request is referred to as the response time. At the time of cloud service agreement, the delay of the response time is agreed. In cloud environments, the least recorded response time is 200*ms*. Mohamed et al. [185] and Beltran et al. [187] took into account the response between 700*m* and 1200*ms*. In the present study, the researchers took into account 1000*ms* in the experiment to ensure the QoS. In the proposed TRP technique, the recorded response time is used to scale the cloud resources in a horizontal way.

SLA Violation: The SLA violation is considered as the time to respond to a user request. The desired response time is 1000*ms* according to the present study experiments. The delay in services than agreed response time leads to the penalty to ASP. In this study, the researcher used the delay of more than 1 second to charge a penalty to application providers.

Cost: The aim of the present study is to minimize the provisioning cost for web application along with QoS in a cloud environment. In literature, most of the articles only considered the renting cost while ignoring the SLA penalty. In the real scenario, the total cost is the sum of the SLA penalty and renting cost. In the present study, the

researcher also considered the total cost according to the real scenario.

5.5.3 Results and Discussion

The effectiveness and performance of the TRP mechanism discussed in this section.

Workload Prediction

The arrival rate of incoming workload is forecast using the prediction model TASM [182]. This model TASM has discussed in detail in Chapter 3. In contrast to the existing time series prediction model, TASM gives better accuracy for web applications workload. The TASM is designed with combination the LR, SVR, and ARIMA models. These models are chosen according to classification on the pattern of the incoming requests. In Chapter 3, the researcher performed the evaluation using 10 minutes short time forecast. The weblogs first compiled in the discrete-time series model. A short prevision of 1 minute was conducted and further applied in this approach in the proposed mechanism TRP. Figure 5.6 shows the prediction result of ClarkNet series with TASM.

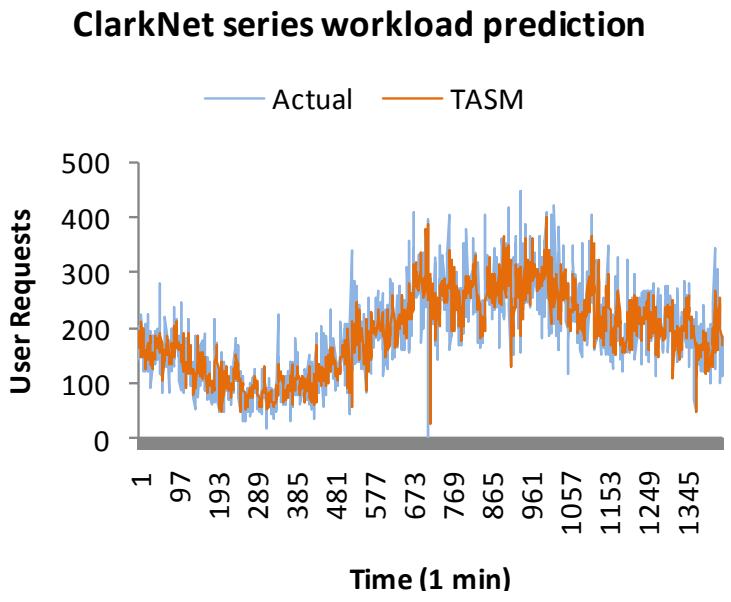

Figure 5.6: Workload prediction of ClarkNet series using TASM prediction model.

$$RMSE = \sqrt{\frac{1}{n}\sum_{s=1}^{n}(actualRequests_s - predictedRequests_s)^2} \tag{5.5}$$

$$MAPE = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{actualRequests_s - predictedRequests_s}{actualRequests_s}\right| \times 100\% \qquad (5.6)$$

Root-Mean-Square-Error (RMSE) in Eq. 5.5 and Mean-Absolute-Percentage-Error (MAPE) in Eq. 5.6 are used to asses accuracy of the prediction models. The TASM prediction model observed value is 42.24 for RMSE and 20.63% for MAPE against the LR values are 64.37 and 34.52% and ARMA values are 51.24 and 32.98% receptively for ClarkNet series.

Triangulation Resource Provisioning Mechanism

In Section 4.2, the researcher explained in detail the TRP mechanism with the help of Figure 5.3. The TRP mechanism is formed with three important parameters related to high-level and low-level details. The experiment evaluation is compared with three different approaches.

The first approach is related to the utilization of resources for deployed VMs (singular approach). In every 1 minute, the monitoring phase records CPU usage, further average CPU utilization is calculated as per the scaling interval. The threshold values are decided based on the type of applications e.g. If average CPU utilization is beyond 80% (upper threshold) than new VM is scaled-out, and a 20% CPU utilization (lower-threshold) leads scale-in decision.

The second approach (double approach) considered in the experiment with additional feature response time with CPU utilization. The delay in response is related to the violation of the SLA agreement. Like the CPU utilization, the response time is also take scaling decision based on threshold values.

In the present study, the 3-D [205] technique based model has designed. Triangulation Resource Provisioning (TRP) added one more parameter that is a number of user request in discrete time series. This parameter is forecast based on the proposed prediction model TASM. The LR, ARIMA and SVR models are selected to forecast based on the classification technique to select the prediction model as per the sliding window pattern. Another contribution of this research is the profit-aware surplus VM selection policy. This help to optimize the QoS along with ASP's profit.

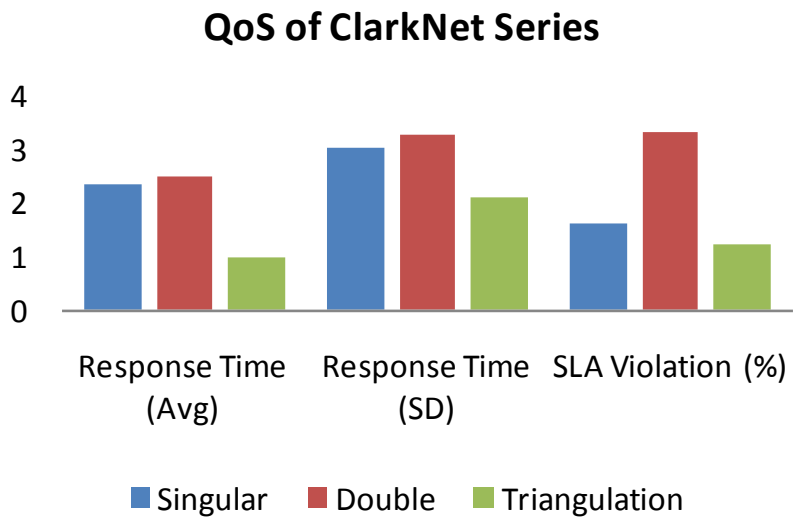

Figure 5.7: The QoS of singular, double and proposed triangulation mechanism for ClarkNet Series.

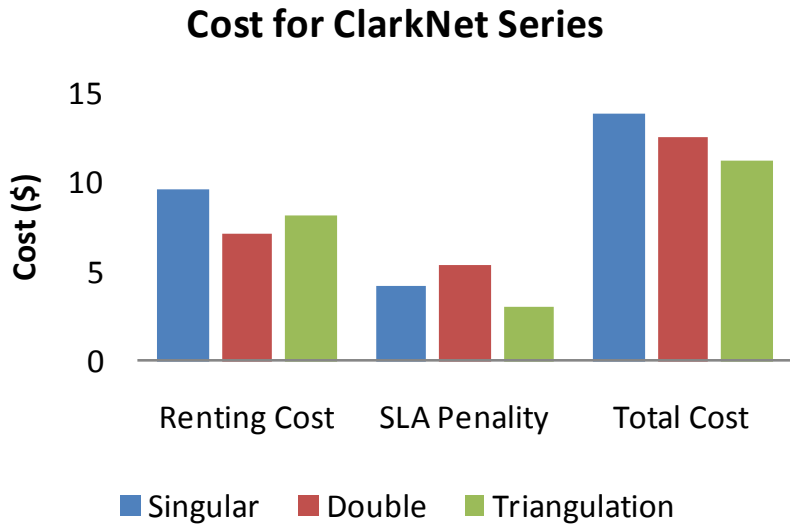

Figure 5.8: The cost of singular, double and proposed triangulation mechanism for ClarkNet Series.

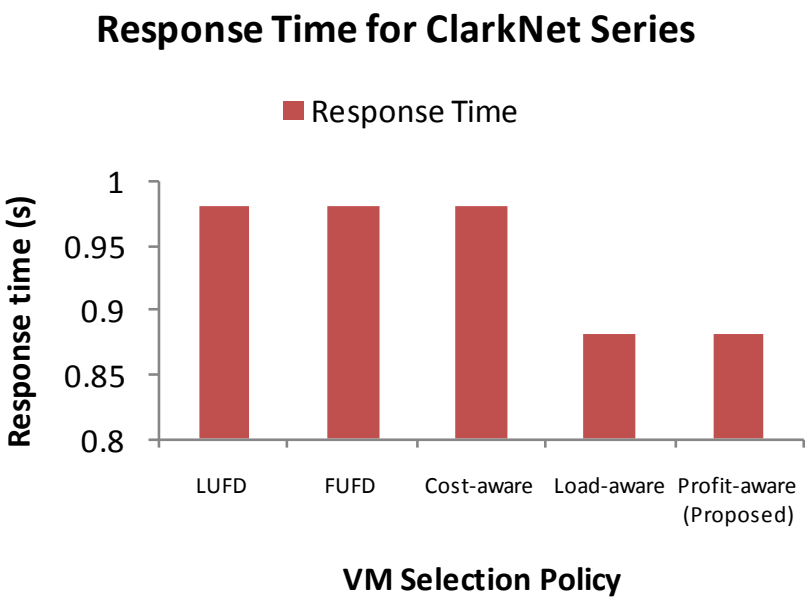

Figure 5.9: The response time of surplus VM selection policies for ClarkNet series.

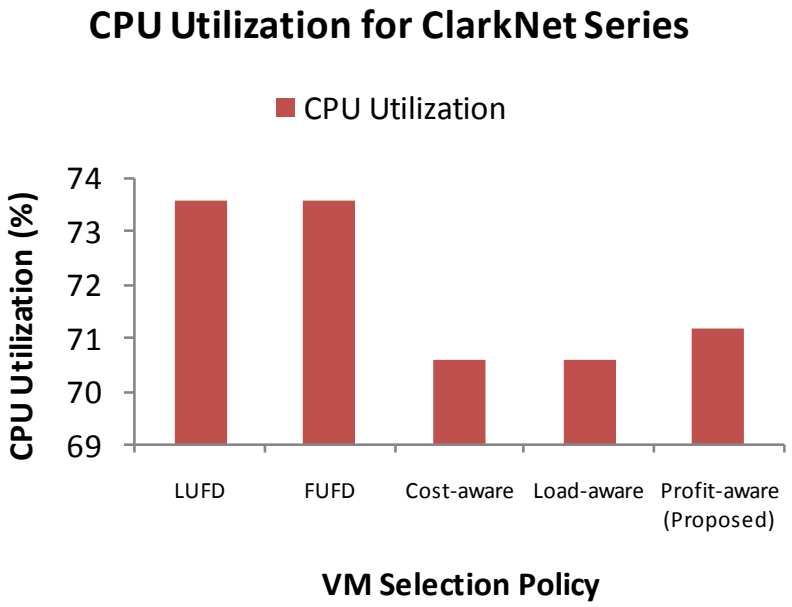

Figure 5.10: The CPU utilization of surplus VM selection policies for ClarkNet series.

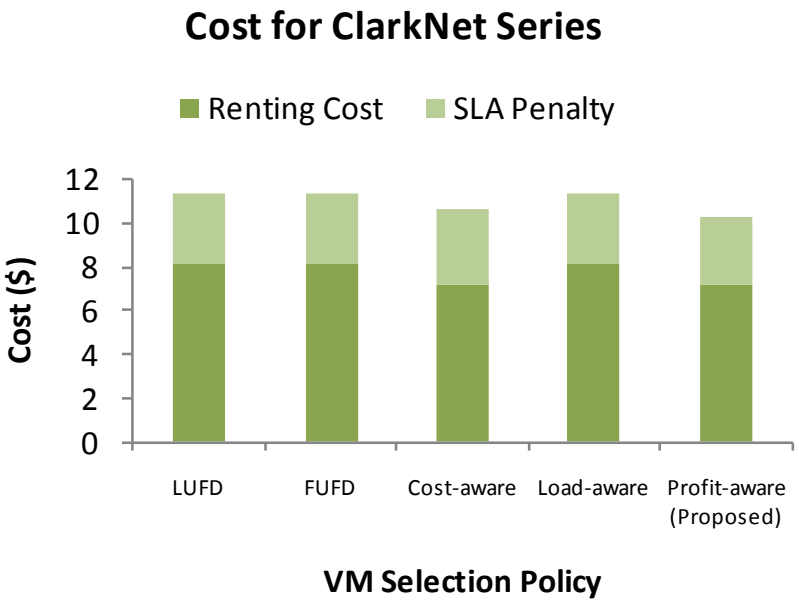

Figure 5.11: The cost of surplus VM selection policies for ClarkNet Series.

The experiment results of resource provisioning mechanisms Singular, Double and Triangulation is shown in Figure 5.7 and Figure 5.8. The response delay is 1 second considered for this experiment as an SLA parameter. The results show that the proposed TRP mechanism is able to provide the response time as per SLA which lead to maintaining the reliability of users on the ASP. The response delay Standard Deviation (SD) shows the QoE of the users. The lower SD value of proposed Triangulation technique depicts that the response delay is close to the desired response time as per SLA agreement. The provisioning cost of the services to ASP are not only renting cost of resources but also the penalty cost, which needs to pay in case of delayed services. The TRP mechanism gives 20% benefit in terms of total provisioning and penalty cost.

Profit-aware Surplus VM Selection Policy

In Amazon EC2, the VM selection for scale-in decision is random. The state-of-the-art policy is developed as per Last Up First Down (LUFD) [202], First Up First Down (FUFD) [131]. Aslanpour et al. also designed the load-aware and cost-aware [132, 205] surplus VM selection policies. In the present study, the experiment is employed the super-professional executor [132]. In this research work, the researcher designed a new profit-aware surplus VM selection technique. The existing cost-aware policy scale-in

the VM with maximum utilization of current billing time period. The issue cost-aware policy is SLA violation due to the number of jobs in the queue need to terminate. In this condition, the SLA penalty faced by the ASP. Another approach in the literature is load-aware policy, which put a more renting cost to the ASP. This technique selects the VM to scale-out, where the number of request in scheduler queue is minimum. This method can choose any VM to de-provision with the minimum workload irrespective of pending service minutes of the billing period. This approach provides the QoS to the end-user but put more burden on ASP in terms of renting cost.

In the present study, the profit-aware surplus VM selection policy identifies the VMs pending service time within the scaling interval. For example, the VMs with a pending service range from 0 to 10 minutes were shortlisted for scaling interval of 10 minutes. The minimum jobs in the scheduler queue of the VMs were then selected for the scale-in action. In every scaling interval, this process is repeated. The proposed VM surplus selection policy shows the significant benefit in experiment results for the ASP and QoS for the end-users.

The ClarkNet web traces are used to emulate the incoming requests in CloudSim toolkit. The experiment is conducted with two types of an executor: Simple and Super Professional (Suprex) [132]. In the first scenario, LUFD and FUFD scaling policies cannot support the Suprex executor, so default executor deployed for these scaling policies. The suprex executor enriched with quarantined VMs technique to simulated for cost-aware, load-aware and proposed profit-aware surplus VM selection policies. The quarantined feature of suprex executor does not implement the scale-in decision immediately, rather it holds the VM till completion of the hourly billing cycle. Thus, it provides the benefit to recall the VM from the quarantined list if VM needed in next scaling interval.

The comparison result of proposed and existing surplus VM selection policies shown in Figure 5.9, Figure 5.10 and Figure 5.11 response time, CPU usage and renting cost respectively. The experiment result showed that the response time of proposed profit-aware and load-aware policies are minimum, thus gives QoS to the end-users. The highest CPU usage observed in LUFD and FUFD scale-in policies. The LUFD and FUFD techniques not able to hold the machine and implement the scaling decision immediately. Due to this, policies gain higher CPU utilization along with higher renting

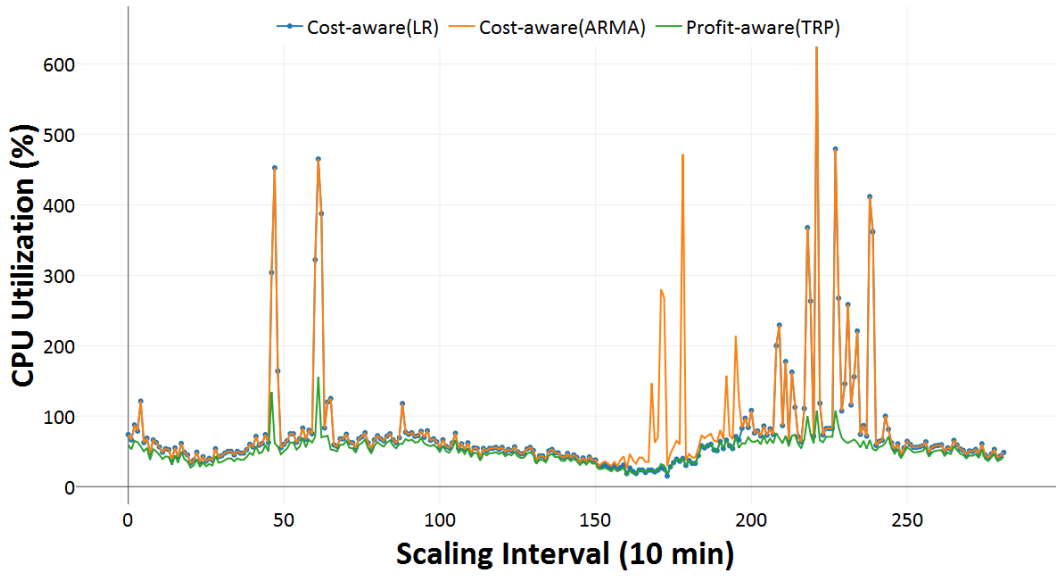

Figure 5.12: The comparison of CPU utilization for ClarkNet Series.

cost also. In total, the proposed profit aware policy benefit the ASP up to 16 percent in cost while giving the QoS experience to the end-users.

Comparison of Resource Provisioning Mechanisms

The cost-aware (LRM) and cost-aware (ARMA) [68] are the two baseline resource provisioning mechanism used to compare the proposed TRP mechanism. Yang et al. [206] applied the Linear Regression (LR) in the cost-aware (LR) provisioning technique. Roy et al. [68] designed the provisioning technique with second-order Autoregressive Moving Average (ARMA) prediction model. In the present study, the proposed profit-aware (TRP) mechanism used the TASM prediction model.

Figure 5.12 described the CPU usage of cost-aware (LRM), cost-aware (ARMA) and proposed profit-aware (TRP) provisioning mechanism for web application in cloud for ClarkNet web trace. The more than 100% CPU usage showed the shortage of resources due to the faulty auto-scaling mechanism. Here, the current resources are not able to handle the incoming workload and ASP has to pay penalty to the end-users. The frequent spikes observed in ARMA and LR resource utilization, this indicates the existing technique is less efficient as compared to proposed TRP mechanism. Thus, the carried out research is successfully able to overcome resource oscillation issues in cloud data centers.

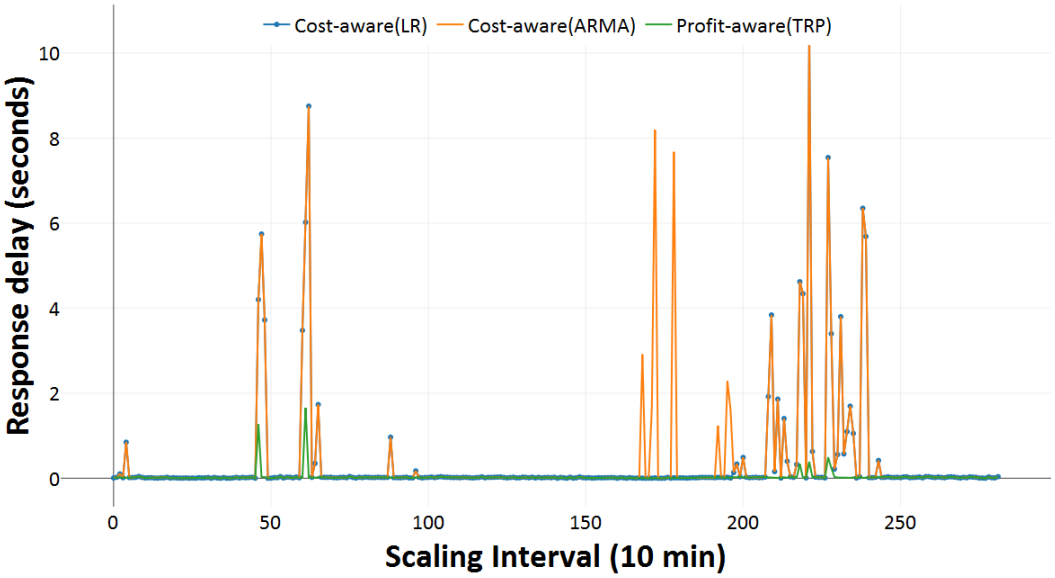

Figure 5.13: The comparison of response delay for ClarkNet Series.

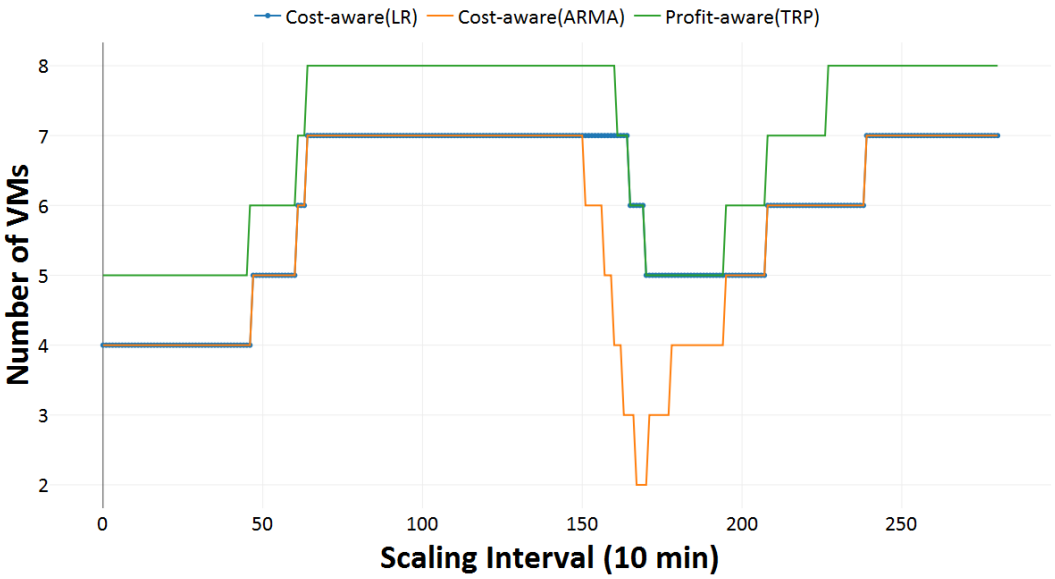

Figure 5.14: The comparison of VM allocation for ClarkNet Series.

The average response delay at each scaling interval of cost-aware (LRM), cost-aware (ARMA) and proposed profit-aware (TRP) mechanisms are shown in Figure 5.13. The response delay is the SLA parameter and key parameter to calculate the SLA penalty. Both under-usage and over-usage of the resource are cost surplus for ASPs. In this experiment, the response delay as per SLA agreement considered as $1s$. Figure 5.13 shows the experiment results for ClarkNet series response delay more than 1 second. The proposed approach experienced very fewer spikes and almost near to 0, this means that the TRP mechanism is a flexible, robust and reliable technique for resource provision come up with the cost and QoS benefits.

The VM allocations for each scaling interval for cost-aware (LRM), cost-aware (ARMA) and proposed profit-aware (TRP) mechanisms are shown in Figure 5.14. The experiment results show that the proposed profit-aware (TRP) approach is more beneficial than cost-aware (LRM) and cost-aware (ARMA) baseline resource provisioning techniques.

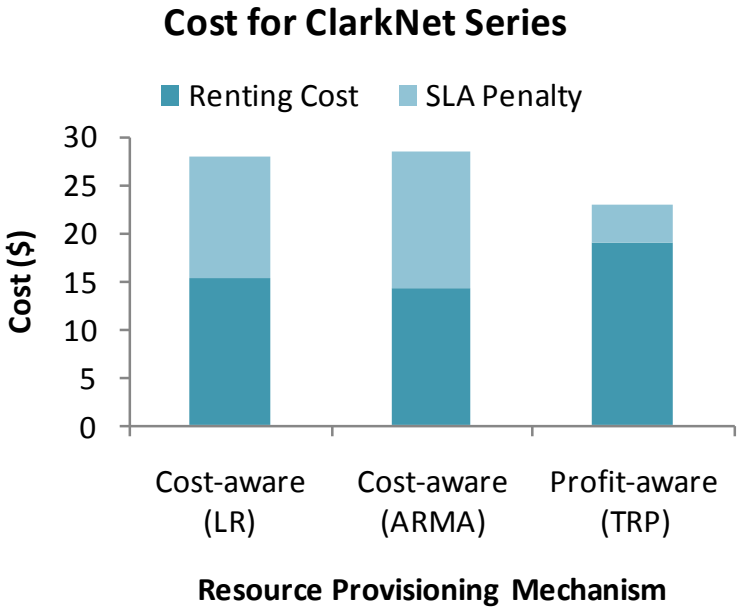

Figure 5.15: The comparison of cost for ClarkNet Series.

The total cost paid by the ASP is SLA penalty to the end-user for delayed services and renting cost to CSP for infrastructure services shown in Figure 5.15. The experiment result shows that the proposed TRP approach is more cost beneficial as compared to other state-of-the-art resource provisioning techniques. The renting cost of TRP

mechanism is higher than other technique but relatively bear minimum SLA penalty. Here, ASP can gain up to 12 percent profit using the proposed resource provisioning techniques.

5.6 Summary

In the present study, resource provisioning technique is developed for web application in cloud infrastructure. The Triangulation Resource Provisioning (TRP) mechanism designed with CPU utilization, response time and workload prediction parameters. In addition, a new profit-aware surplus VM selection policy designed for scale-in decision. The TRP mechanism designed under MAPE control loop. The monitoring phase collects the infrastructure and user level parameters. The analysis phase processed the information with TASM prediction model, average CPU usage and average response time. The planning phase heuristic is used to take the scaling decision. The execution phase designed with super-professional executor for profit-aware surplus VM selection policy. The proposed technique select the appropriate VM to scale-in from the surplus VM list, so that it could save the cost and maintain the desired QoS as per SLA. The ClarkNet workload traces emulated the incoming traffic. R tool and CloudSim enhanced with auto-scaling libraries used to perform a comparison between existing and proposed resource provisioning techniques. The proposed provisioning approach with profit-aware surplus VM selection policy is a profit-aware approach can provide the application providers to save the overall provisioning cost while giving the QoS to the end-users for web application in the cloud data centers.

CHAPTER 6

CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion and Discussion

Cloud computing provides the resources in a pay-as-you-go manner to the users as per the real-time demand of the applications. This feature eliminates the requirement of local infrastructure and users can concentrate on their main business. The web application providers have been attracted towards this appealing feature.

The application providers have been deployed the applications on the cloud data centers. The cloud computing brings advantages for web application providers like higher availability, QoS, cost efficiency and reliability. However, the resource provisioning of web application have some pending challenges.

In this thesis, the researcher divided the challenges into three aspects: 1) workload prediction of web applications, 2) auto-scaling of web application in the cloud, and 3) resource provisioning to raise profit with QoS. The researcher performed a methodological literature review, proposed solutions, developed the mechanisms, and simulate the techniques to escalate the current state-of-the-art in these areas. In particular, Chapter 1 explained the detail of the objectives of this thesis and described the contribution of the present study. It also defines the structure of the thesis.

In Chapter 2, a methodological survey carried for the existing research on the prob-

lems addressed in this research work. The survey starts with the origin of autonomous computing. The detail investigation has been carried for elastic applications, capacity management and QoS-aware requirement of web applications in cloud data centers. A comprehensive taxonomy on auto-scaling techniques is proposed and compared the literature articles. The resource provisioning technique is discussed in detail in each auto-scaling category. The researcher also identified the challenges in resource provisioning of cloud applications.

Chapter 3 presented the efficient time series prediction model specifically for web applications in the cloud. The name of the developed prediction model is Technocrat ARIMA and SVR Model (TASM). It is the combination of three prediction models along with the workload classification approach. There are various types of workload patterns presented in the workload of web applications. In literature, there is no general method that fits with all types of patterns present in the workload. The classification of workload performed on the basis of $l2-norm$, average-rate-of-change and linearity test. A mechanism is designed to configure the model's parameters with residual testing. In addition to existing models, it is able to predict the workload with seasonal and non-seasonal patterns. A new cloud architecture designed with the proposed prediction model named as Technocrat Cloud Provisioning Architecture. The researcher developed this model using *R* programming language. The ClarkNet and NASA weblogs converted to a discrete time series for performance evaluation. The error rate in the existing and proposed model tested with the performance metrics such as MAE, MSE, RMSE, and MAPE. The platform error calculated for over-provisioning and under-provisioning of the resources. The experimental results confirmed the higher efficiency of TASM for web application in the cloud environment. It makes a significant improvement in resource oscillation problem in the cloud applications.

Chapter 4 proposed a Robust Hybrid Auto-Scaler (RHAS) technique to solve the web application provider's issue of uncertain scaling decisions due to irregular incoming requests. The technique is designed on the basis of Monitor-Analyze-Plan-Execute and knowledge (MAPE-K) architecture. The monitor phase collects the data for user requests and resource utilization. The hybrid analysis phase is designed to calculate the future incoming request with TASM prediction model, average CPU utilization and average response time. These values further input to the hybrid planning phase,

which is designed with the combination of reactive and proactive auto-scaling methods. CloudSim toolkit is extended with auto-scaling libraries and $R-caller$ library for the simulation of the proposed RHAS auto-scaling technique. The efficiency of RHAS is demonstrated through simulation experiments. The simulation results proved that RHAS reduces the renting cost and SLA violation with a fair amount of CPU utilization.

Chapter 5 presented the resource provisioning mechanism with a comprehensive work of workload prediction, auto-scaling technique. The proposed technique is designed to maximize the profit of the application provider and also ensure the QoS to the end-user. Triangulation Resource Provisioning (TRP) mechanism designed using workload prediction, CPU utilization and response time. This technique focused on the execution phase of MAPE-K architecture, where the decision of planning phase collides with infrastructure level constraints. The super-professional executors gave the facility to quarantined the spare capacity. The scale-in decision in horizontal scaling needs to select an in-service Virtual Machine (VM). The profit-aware surplus VM selection policy developed to select the VM with the least load in the current scaling interval. CloudSim toolkit is enhanced with classes of the execution phase. The real workload of ClarkNet series is used in the experimental evaluation. The simulation results demonstrated that proposed TRP and Profit-aware Surplus VM selection policy stable the CPU utilization, enhance the renting cost for fair QoS and reduce the SLA penalty. The total cost to the application providers is the sum of renting cost and SLA penalty. Therefore, the TRP mechanism is superior in terms of cost and QoS. The application providers reap more profit and users get QoS for the web applications in the cloud environment.

6.2 Future Directions

Although many research efforts have researched the resource provisioning problem for cloud applications. There are still scope of improvement and research gaps to be investigated for the multi-tier web applications. The following sections describe important future directions in this field.

6.2.1 Monitoring Tools

Cost effective monitoring tools are required to implement, which explore the hidden parameters such as cache memory, service time and types of request (e.g. Compute intensive and data intensive). Application and workload-specific monitoring interval can further improve the auto-scaler results.

6.2.2 Pricing Model

The companies such as Amazon, Google and Microsoft have their different pricing model. Most of the researchers considered auto-scaling techniques with on-demand resources while considering the unlimited resources. Other types of pricing are also introduced e.g. Spot instances by the Amazon. The cost of such resources is very less as compared to on-demand resources. Very few authors start working on the spot instance for web applications, so the area is very immature and have different research challenges. Auto-scaling techniques for spot-instances using reactive and proactive techniques give more benefit to application providers and clients.

6.2.3 Resource Allocation

The information on data center resources plays a significant role in resource provisioning decisions. Some parameters such as CPU, RAM, Disk are well-known factors but some parameters such as cache memory, network bandwidth, and fault tolerance are least considered in the literature. VMs for input workload for different tiers of the web application is the future scope of many articles. SLA based resource allocation with different QoS parameters needs improvement. It can be further improved by extending the queuing network model.

6.2.4 Horizontal and Vertical Scaling

Horizontal scaling is used in most of the articles. Operating system and cloud architecture support horizontal scaling. The vertical scaling is easier in cloud infrastructure and gives better cost benefit. Hypervisors and operating system support could be enhanced for the vertical scaling. Energy and cost-effective VMs allocation and consolidated with migration are the future areas for research.

6.2.5 Workload Predictor

The existing workload predictors are considered the historical workload. Flash workload is hard to predict from the past workload. As growth in the online data mining and deep learning techniques, real-time data can be used to avoid the sudden burst condition in the data center. Web applications face this problem due to any hackers attack (DDOS) or flash event. Categorization of application will be more helpful to handle the flash crowd. Spot instances can be more effective in terms of cost optimization to serve flash workload.

6.2.6 Multi-cloud Auto-scaling

Multiple cloud providers are supporting multi-tier web architecture. Cloud providers individually proved the reliability of their services. Multi-cloud architecture for web applications needs to work upon. The cloud provider can be selected by considering the application feature as the type of application, input workload, geographical area, etc. The reliability-aware auto-scaling in the multi-cloud environment from factorizing the various parameters guarantee the trust of users.

6.2.7 Energy-aware Auto-scaling

The existing work mostly focused on cost optimization and QoS requirement. The data center is also facing environmental issues also. Solar data centers and renewable energy resources in the data center can reduce the carbon problem. The carbon and energy-aware auto-scaling provide the preference to the environmental friendly data centers in a single and multi-cloud environment. The weather condition and maximum solar power generating data centers give the highest priority for resource allocation.

6.2.8 Bin-packing Auto-scaling

The bin packing approach offers potential research topics in auto-scaling of web applications in cloud computing. The size of bins could vary, which makes this approach more robust. Resource provisioning of bins is lightweight techniques and gives cost benefits to the providers.

6.3 Final Remarks

Cloud computing has enough potential to save the cost and provide quality service for multi-tier web applications. The competitive market of the cloud opens up the challenges and opportunities to solve these challenges. In this thesis, the researcher traversed the problems of the application providers for proactive resource management of the multi-tier web applications in the cloud. The proposed resource provisioning mechanisms in this research are influential in developing the profit-oriented middle-ware for application providers. It will accredit the resource provisioning with cost saving, QoE satisfaction and fair resource utilization in the cloud infrastructure.

REFERENCES

[1] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, 2014.

[2] K. B. Douglas, "Web services and service-oriented architectures: the savvy manager's guide," 2003.

[3] V. A. Almeida and D. A. Menascé, "Capacity planning an essential tool for managing web services," *IT professional*, vol. 4, no. 4, pp. 33–38, 2002.

[4] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and qos in cloud computing environments," in *Parallel Processing (ICPP), 2011 International Conference on*, pp. 295–304, IEEE, 2011.

[5] W. Iqbal, M. N. Dailey, D. Carrera, and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871–879, 2011.

[6] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[7] J. W. Cohen, *The single server queue*, vol. 8. Elsevier, 2012.

[8] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as

the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[9] A. Huth and J. Cebula, "The basics of cloud computing," *United States Computer*, 2011.

[10] KPMG, "2014 cloud survey report," *KPMG, Report*, 2014.

[11] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, *et al.*, "Above the clouds: A berkeley view of cloud computing," 2009.

[13] J. Lango, "Toward software-defined slas," *Communications of the ACM*, vol. 57, no. 1, pp. 54–60, 2014.

[14] J. Varia and S. Mathew, "Overview of amazon web services," *Amazon Web Services*, 2014.

[15] T. Lorido-Botrán, J. Miguel-Alonso, and J. A. Lozano, "Auto-scaling techniques for elastic applications in cloud environments," *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09*, vol. 12, p. 2012, 2012.

[16] H. Kim, M. Parashar, D. J. Foran, and L. Yang, "Investigating the use of autonomic cloudbursts for high-throughput medical image registration," in *2009 10th IEEE/ACM International Conference on Grid Computing*, pp. 34–41, IEEE, 2009.

[17] "An architectural blueprint for autonomic computing," tech. rep., IBM, June 2005.

[18] M. Rahman, R. Ranjan, R. Buyya, and B. Benatallah, "A taxonomy and survey on autonomic management of applications in grid computing environments," *Concurrency and computation: practice and experience*, vol. 23, no. 16, pp. 1990–2019, 2011.

[19] W. S. Mandak, C. A. Stowell, *et al.*, *Dynamic assembly for system adaptability, dependability and assurance (DASADA) project analysis*. PhD thesis, 2001.

[20] P. Horn, "Autonomic computing: Ibm\'s perspective on the state of information technology," 2001.

[21] I. Tivoli, "Autonomic computing policy language," *Tutorial, IBM Corp*, 2005.

[22] M. Parashar and S. Hariri, "Autonomic grid computing," *Proceedings of International Conference on Autonomic Computing, U.S.A.*, pp. 1–10, May 2005.

[23] H. Kim, Y. El-Khamra, S. Jha, and M. Parashar, "An autonomic approach to integrated hpc grid and cloud usage," in *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pp. 366–373, IEEE, 2009.

[24] M. Maurer, I. Breskovic, V. C. Emeakaroha, and I. Brandic, "Revealing the mape loop for the autonomic management of cloud infrastructures," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pp. 147–152, IEEE, 2011.

[25] H. Ghanbari, B. Simmons, M. Litoiu, and G. Iszlai, "Exploring alternative approaches to implement an elasticity policy," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 716–723, IEEE, 2011.

[26] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 423–430, IEEE, 2012.

[27] J. Yang, T. Yu, L. Jian, J. Qiu, and Y. Li, "An extreme automation framework for scaling cloud applications," *IBM Journal of Research and Development*, vol. 55, no. 6, pp. 8–1, 2011.

[28] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pp. 456–463, IEEE, 2010.

[29] RackSpace, "RightScale public cloud infrastructure." `https://www.rackspace.com/`, 2019. [Online; accessed 10-April-2019].

[30] J. Kupferman, J. Silverman, P. Jara, and J. Browne, "Scaling into the cloud," *CS270-advanced operating systems*, 2009.

[31] T. Lorido-Botran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.

[32] R. R. U. B. System, "http://rubis.ow2.org/, 2009.," *[Online Accessed: 20-02-2016].*

[33] TPC-W, "http://www.tpc.org/tpcw/default.asp, 2012.," *[Online Accessed: 20-02-2016].*

[34] C. P. by Rad Lab Group., "http://radlab.cs.berkeley.edu/wiki/projects/cloudstone/, 2012.," *[Online Accessed: 20-02-2016].*

[35] S. T. httperf HTTP load generator., "http://www.spec.org/web2009/, 2012.," *[Online Accessed: 19-02-2016].*

[36] TPC-C., "http://www.tpc.org/tpcc/default.asp/, 2012.," *[Online Accessed: 19-02-2016].*

[37] R. B. B. Benchmark., "http://www.tpc.org/tpcc/default.asp/, 2012.," *[Online Accessed: 19-02-2016].*

[38] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98–115, 2015.

[39] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *Algorithms and Architectures for Parallel Processing*, pp. 371–384, Springer, 2011.

[40] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[41] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. G. Greenberg, and Y.-M. Wang, "Webprophet: Automating performance prediction for web services.," in *NSDI*, vol. 10, pp. 143–158, 2010.

[42] A. Li, X. Zong, M. Zhang, S. Kandula, and X. Yang, "Cloudprophet: predicting web application performance in the cloud," *ACM SIGCOMM Poster*, 2011.

[43] S. Spinner, G. Casale, F. Brosig, and S. Kounev, "Evaluating approaches to resource demand estimation," *Performance Evaluation*, vol. 92, pp. 51–71, 2015.

[44] A. E. C. C. A. AutoScaling., "http://aws.amazon.com/autoscaling/.," *[Online Accessed: 19-02-2016].*

[45] R. B. B. B. A. Scaling., "http:// support.rightscale.com/03-tutorials/02-aws/02-website edition/set up autoscaling using voting tags.," *[Online Accessed: 19-02-2016].*

[46] J. Guitart, J. Torres, and E. Ayguadé, "A survey on performance management for internet applications," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 1, pp. 68–106, 2010.

[47] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424–440, 2014.

[48] P. Singh, G. B. Singh, and K. Jyoti, "A study on resource provisioning of multi-tier web applications in cloud computing," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 799–802, IEEE, 2015.

[49] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: a taxonomy and survey," *arXiv preprint arXiv:1609.09224*, 2016.

[50] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "A cost-aware elasticity provisioning system for the cloud," in *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pp. 559–570, IEEE, 2011.

[51] H. Fernandez, G. Pierre, and T. Kielmann, "Autoscaling web applications in heterogeneous cloud infrastructures," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 195–204, IEEE, 2014.

[52] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "Autoscale: Dynamic, robust capacity management for multi-tier data centers," *ACM Transactions on Computer Systems (TOCS)*, vol. 30, no. 4, p. 14, 2012.

[53] C. Qu, R. N. Calheiros, and R. Buyya, "A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances," *Journal of Network and Computer Applications*, vol. 65, pp. 167–180, 2016.

[54] N. Vasić, D. Novaković, S. Miučin, D. Kostić, and R. Bianchini, "Dejavu: accelerating resource allocation in virtualized environments," in *ACM SIGARCH computer architecture news*, vol. 40, pp. 423–436, ACM, 2012.

[55] H. Nguyen, Z. Shen, X. Gu, S. Subbiah, and J. Wilkes, "Agile: Elastic distributed resource scaling for infrastructure-as-a-service.," in *ICAC*, vol. 13, pp. 69–82, 2013.

[56] J. Dejun, G. Pierre, and C.-H. Chi, "Resource provisioning of web applications in heterogeneous clouds," in *Proceedings of the 2nd USENIX conference on Web application development*, pp. 5–5, USENIX Association, 2011.

[57] T. Patikirikorala and A. Colman, "Feedback controllers in the cloud," in *Proceedings of APSEC*, 2010.

[58] H. C. Lim, S. Babu, J. S. Chase, and S. S. Parekh, "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, 2009.

[59] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," in *Proceedings of the 7th international conference on Autonomic computing*, pp. 1–10, ACM, 2010.

[60] S.-M. Park and M. Humphrey, "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, 2009.

[61] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pp. 304–307, ACM, 2010.

[62] A. Ali-Eldin, J. Tordsson, and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *2012 IEEE Network Operations and Management Symposium*, pp. 204–212, IEEE, 2012.

[63] A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth, "Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control," in *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pp. 31–40, ACM, 2012.

[64] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proceedings of the 4th ACM European conference on Computer systems*, pp. 13–26, ACM, 2009.

[65] P. Bodık, R. Griffith, C. Sutton, A. Fox, M. Jordan, and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters," in *Proceedings of the 2009 conference on Hot topics in cloud computing*, pp. 12–12, 2009.

[66] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters," in *Proceedings of the 6th international conference on Autonomic computing*, pp. 117–126, ACM, 2009.

[67] A. Gambi and G. Toffetti, "Modeling cloud performance with kriging," in *Proceedings of the 34th International Conference on Software Engineering*, pp. 1439–1440, IEEE Press, 2012.

[68] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 500–507, IEEE, 2011.

[69] L. Wang, J. Xu, M. Zhao, Y. Tu, and J. A. Fortes, "Fuzzy modeling based resource management for virtualized database systems," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 32–42, IEEE, 2011.

[70] S. Farokhi, P. Jamshidi, E. B. Lakew, I. Brandic, and E. Elmroth, "A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach," *Future Generation Computer Systems*, vol. 65, pp. 57–72, 2016.

[71] P. Lama and X. Zhou, "Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 8, no. 2, p. 9, 2013.

[72] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, "On the use of fuzzy modeling in virtualized data center management," in *Fourth International Conference on Autonomic Computing (ICAC'07)*, pp. 25–25, IEEE, 2007.

[73] T. Heinze, V. Pappalardo, Z. Jerzak, and C. Fetzer, "Auto-scaling techniques for elastic data stream processing," in *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pp. 296–302, IEEE, 2014.

[74] M. Fallah, M. G. Arani, and M. Maeen, "Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment," *International Journal of Computer Applications*, vol. 113, no. 2, 2015.

[75] D. Grimaldi, A. Pescape, A. Salvi, V. Persico, *et al.*, "A fuzzy approach based on heterogeneous metrics for scaling out public clouds," *IEEE Transactions on Parallel and Distributed Systems*, 2017.

[76] P. Lama and X. Zhou, "Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters," in *Quality of Service, 2009. IWQoS. 17th International Workshop on*, pp. 1–9, IEEE, 2009.

[77] S. Frey, C. Lüthje, C. Reich, and N. Clarke, "Cloud qos scaling by fuzzy logic," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 343–348, IEEE, 2014.

[78] P. Jamshidi, C. Pahl, and N. C. Mendonça, "Managing uncertainty in autonomic cloud elasticity controllers," *IEEE Cloud Computing*, vol. 3, no. 3, pp. 50–60, 2016.

[79] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, vol. 135. MIT Press Cambridge, 1998.

[80] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in *2006 IEEE International Conference on Autonomic Computing*, pp. 65–73, IEEE, 2006.

[81] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, pp. 67–74, 2011.

[82] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, "Vconf: a reinforcement learning approach to virtual machines auto-configuration," in *Proceedings of the 6th international conference on Autonomic computing*, pp. 137–146, ACM, 2009.

[83] C.-Z. Xu, J. Rao, and X. Bu, "Url: A unified reinforcement learning approach for autonomic cloud management," *Journal of Parallel and Distributed Computing*, vol. 72, no. 2, pp. 95–105, 2012.

[84] J. Rao, X. Bu, C.-Z. Xu, and K. Wang, "A distributed self-learning approach for elastic provisioning of virtualized cloud resources," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 45–54, IEEE, 2011.

[85] X. Bu, J. Rao, and C.-Z. Xu, "Coordinated self-configuration of virtual machines and appliances using a model-free learning approach," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 4, pp. 681–690, 2013.

[86] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Cloud resource auto-scaling system based on hidden markov model (hmm)," in *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pp. 124–127, IEEE, 2014.

[87] C. Liu, C. Liu, Y. Shang, S. Chen, B. Cheng, and J. Chen, "An adaptive prediction approach based on workload pattern discrimination in the cloud," *Journal of Network and Computer Applications*, vol. 80, pp. 35–44, 2017.

[88] E. Barrett, E. Howley, and J. Duggan, "Applying reinforcement learning towards automating resource allocation and application scalability in the cloud," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 12, pp. 1656–1674, 2013.

[89] X. Dutreilh, A. Moreau, J. Malenfant, N. Rivierre, and I. Truck, "From data center resource allocation to control theory and back," in *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 410–417, IEEE, 2010.

[90] D. G. Kendall, "Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain," *The Annals of Mathematical Statistics*, pp. 338–354, 1953.

[91] J. Keilson and L. Servi, "A distributional form of little's law," *Operations Research Letters*, vol. 7, no. 5, pp. 223–227, 1988.

[92] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 3, no. 1, p. 1, 2008.

[93] Q. Zhang, L. Cherkasova, and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," in *Fourth International Conference on Autonomic Computing (ICAC'07)*, pp. 27–27, IEEE, 2007.

[94] R. Han, M. M. Ghanem, L. Guo, Y. Guo, and M. Osmond, "Enabling cost-aware and adaptive elasticity of multi-tier cloud applications," *Future Generation Computer Systems*, vol. 32, pp. 82–98, 2014.

[95] D. A. Bacigalupo, J. van Hemert, A. Usmani, D. N. Dillenberger, G. B. Wills, and S. A. Jarvis, "Resource management of enterprise cloud systems using layered queuing and historical performance models," in *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pp. 1–8, IEEE, 2010.

[96] D. Villela, P. Pradhan, and D. Rubenstein, "Provisioning servers in the application tier for e-commerce systems," *ACM Transactions on Internet Technology (TOIT)*, vol. 7, no. 1, p. 7, 2007.

[97] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[98] T. Vondra and J. Šedivỳ, "Cloud autoscaling simulation based on queueing network model," *Simulation Modelling Practice and Theory*, vol. 70, pp. 83–100, 2017.

[99] A. Gandhi, P. Dube, A. Karve, A. Kochut, and L. Zhang, "Adaptive, model-driven autoscaling for cloud applications," in *11th International Conference on Autonomic Computing (ICAC 14)*, pp. 57–64, 2014.

[100] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[101] P. Koperek and W. Funika, "Dynamic business metrics-driven resource provisioning in cloud environments," in *International Conference on Parallel Processing and Applied Mathematics*, pp. 171–180, Springer, 2011.

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

[103] M. Z. Hasan, E. Magana, A. Clemm, L. Tucker, and S. L. D. Gudreddi, "Integrated and autonomic cloud resource scaling," in *2012 IEEE Network Operations and Management Symposium*, pp. 1327–1334, IEEE, 2012.

[104] I. Mahallat, "Astaw: Auto-scaling threshold-based approach for web application in cloud computing environment," 2015.

[105] B. Simmons, H. Ghanbari, M. Litoiu, and G. Iszlai, "Managing a saas application in the cloud using paas policy sets and a strategy-tree," in *Proceedings of the 7th*

International Conference on Network and Services Management, pp. 343–347, International Federation for Information Processing, 2011.

[106] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "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, 2009.

[107] T. C. Chieu, A. Mohindra, and A. A. Karve, "Scalability and performance of web applications in a compute cloud," in *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, pp. 317–323, IEEE, 2011.

[108] T. Lorido-Botrcn, J. Miguel-Alonso, and J. A. Lozano, "Comparison of auto-scaling techniques for cloud environments," 2013.

[109] N. Grozev and R. Buyya, "Multi-cloud provisioning and load distribution for three-tier applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 9, no. 3, p. 13, 2014.

[110] E. Casalicchio and L. Silvestri, "Autonomic management of cloud-based systems: the service provider perspective," in *Computer and Information Sciences III*, pp. 39–47, Springer, 2013.

[111] M. Maurer, I. Brandic, and R. Sakellariou, "Enacting slas in clouds using rules," in *European Conference on Parallel Processing*, pp. 455–466, Springer, 2011.

[112] S. Khatua, A. Ghosh, and N. Mukherjee, "Optimizing the utilization of virtual resources in cloud environment," in *2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pp. 82–87, IEEE, 2010.

[113] R. G. Brown and R. F. Meyer, "The fundamental theorem of exponential smoothing," *Operations Research*, vol. 9, no. 5, pp. 673–685, 1961.

[114] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms second edition," 2001.

[115] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management*, pp. 9–16, IEEE, 2010.

[116] J. Huang, C. Li, and J. Yu, "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, 2012.

[117] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, and L. Yuan, "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, 2010.

[118] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *Quality of Service—IWQoS 2003*, pp. 381–398, Springer, 2003.

[119] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services.," in *NSDI*, vol. 8, pp. 337–350, 2008.

[120] W. Fang, Z. Lu, J. Wu, and Z. Cao, "Rpps: a novel resource prediction and provisioning scheme in cloud data center," in *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pp. 609–616, IEEE, 2012.

[121] M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth, "A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, p. 6, ACM, 2013.

[122] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications' qos," *Cloud Computing, IEEE Transactions on*, vol. 3, no. 4, pp. 449–458, 2015.

[123] V. R. Messias, J. C. Estrella, R. Ehlers, M. J. Santana, R. C. Santana, and S. Reiff-Marganiec, "Combining time series prediction models using genetic algorithm to

autoscaling web applications hosted in the cloud infrastructure," *Neural Computing and Applications*, pp. 1–24, 2016.

[124] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pp. 1–12, IEEE, 2011.

[125] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[126] R. Prodan and V. Nae, "Prediction-based real-time resource provisioning for massively multiplayer online games," *Future Generation Computer Systems*, vol. 25, no. 7, pp. 785–793, 2009.

[127] S. Dutta, S. Gera, A. Verma, and B. Viswanathan, "Smartscale: Automatic application scaling in enterprise clouds," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 221–228, IEEE, 2012.

[128] E. Caron, F. Desprez, and A. Muresan, "Pattern matching based forecast of non-periodic repetitive behavior for cloud clients," *Journal of Grid Computing*, vol. 9, no. 1, pp. 49–64, 2011.

[129] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, p. 5, ACM, 2011.

[130] R. E. De Grande, A. Boukerche, and R. Alkharboush, "Time series-oriented load prediction model and migration policies for distributed simulation systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 215–229, 2017.

[131] M. S. Aslanpour and S. E. Dashti, "Proactive auto-scaling algorithm (pasa) for cloud application," *International Journal of Grid and High Performance Computing (IJGHPC)*, vol. 9, no. 3, pp. 1–16, 2017.

[132] M. S. Aslanpour, M. Ghobaei-Arani, and A. N. Toosi, "Auto-scaling web applications in clouds: a cost-aware approach," *Journal of Network and Computer Applications*, vol. 95, pp. 26–41, 2017.

[133] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback control of computing systems*. John Wiley & Sons, 2004.

[134] W. Lloyd, S. Pallickara, O. David, J. Lyon, M. Arabi, and K. Rojas, "Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: Towards performance modeling," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1254–1264, 2013.

[135] P. Xiong, Z. Wang, G. Jung, and C. Pu, "Study on performance management and application behavior in virtualized environment," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pp. 841–844, IEEE, 2010.

[136] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 33, pp. 291–302, ACM, 2005.

[137] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed systems meet economics: Pricing in the cloud.," *HotCloud*, vol. 10, pp. 1–6, 2010.

[138] S. Ibrahim, B. He, and H. Jin, "Towards pay-as-you-consume cloud computing," in *Services Computing (SCC), 2011 IEEE International Conference on*, pp. 370–377, IEEE, 2011.

[139] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "Above the clouds: A berkeley view of cloud computing," *Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS*, vol. 28, no. 13, p. 2009, 2009.

[140] A. M. C. Halavais, *The Slashdot effect: analysis of a large-scale public conversation on the World Wide Web*. PhD thesis, 2001.

[141] J. Panneerselvam, L. Liu, N. Antonopoulos, and Y. Bo, "Workload analysis for the scope of user demand prediction model evaluations in cloud environments,"

in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 883–889, IEEE Computer Society, 2014.

[142] H. Zhang, G. Jiang, K. Yoshihira, and H. Chen, "Proactive workload management in hybrid cloud computing," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 90–100, 2014.

[143] A. A. Eldin, A. Rezaie, A. Mehta, S. Razroev, S. S. de Luna, O. Seleznjev, J. Tordsson, and E. Elmroth, "How will your workload look like in 6 years? analyzing wikimedia's workload," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 349–354, IEEE, 2014.

[144] J. Patel, V. Jindal, I.-L. Yen, F. Bastani, J. Xu, and P. Garraghan, "Workload estimation for improving resource management decisions in the cloud," in *Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on*, pp. 25–32, IEEE, 2015.

[145] K. Wang, M. Lin, F. Ciucu, A. Wierman, and C. Lin, "Characterizing the impact of the workload on the value of dynamic resizing in data centers," *Performance Evaluation*, vol. 85, pp. 1–18, 2015.

[146] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "Burse: A bursty and self-similar workload generator for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 668–680, 2015.

[147] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: a survey revisited," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 48, 2016.

[148] N. Bonvin, T. G. Papaioannou, and K. Aberer, "Autonomic sla-driven provisioning for cloud applications," in *Proceedings of the 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing*, pp. 434–443, IEEE Computer Society, 2011.

[149] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, "Intelligent workload factoring for a hybrid cloud computing model," in *Services-I, 2009 World Conference on*, pp. 701–708, IEEE, 2009.

[150] Y. Jiang, C.-S. Perng, T. Li, and R. N. Chang, "Cloud analytics for capacity planning and instant vm provisioning," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 312–325, 2013.

[151] N. I. Sapankevych and R. Sankar, "Time series prediction using support vector machines: a survey," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, 2009.

[152] A. A. Bankole and S. A. Ajila, "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, 2013.

[153] P. S. Doshi, M. Goel, A. Agarwal, and K. Punjabi, "Performance provisioning using machine learning based automated workload classification," 2018. US Patent App. 15/257,491.

[154] T. Teräsvirta, C.-F. Lin, and C. W. Granger, "Power of the neural network linearity test," *Journal of Time Series Analysis*, vol. 14, no. 2, pp. 209–220, 1993.

[155] R. C. Team *et al.*, "R: A language and environment for statistical computing," 2013.

[156] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2014.

[157] S. R. Gunn *et al.*, "Support vector machines for classification and regression," *ISIS technical report*, vol. 14, no. 1, pp. 5–16, 1998.

[158] C.-H. Wu, C.-C. Wei, D.-C. Su, M.-H. Chang, and J.-M. Ho, "Travel time prediction with support vector regression," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 2, pp. 1438–1442, IEEE, 2003.

[159] X. Yan and X. Su, *Linear regression analysis: theory and computing*. World Scientific, 2009.

[160] G. A. Seber and A. J. Lee, *Linear regression analysis*, vol. 329. John Wiley & Sons, 2012.

[161] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting methods and applications*. John wiley & sons, 2008.

[162] G. M. Ljung and G. E. Box, "On a measure of lack of fit in time series models," *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978.

[163] S. Balbach, "Clarknet web server logs," *URL http://ita. ee. lbl. gov/html/contrib/ClarkNet-HTTP. html, accessed: 25 Feb*, 2018.

[164] J. Dumoulin, "Nasa web server logs," *URL http://ita. ee. lbl. gov/html/contrib/NASA-HTTP. html, accessed: 25 Feb*, 2018.

[165] Y. Xiaofang and W. Yaonan, "Parameter selection of support vector machine for function approximation based on chaos optimization," *Journal of Systems Engineering and Electronics*, vol. 19, no. 1, pp. 191–197, 2008.

[166] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *annals of telecommunications-annales des télécommunications*, vol. 70, no. 7-8, pp. 289–309, 2015.

[167] A. EC2", ""spot instances"," 2018.

[168] Y. Shen, H. Chen, L. Shen, C. Mei, and X. Pu, "Cost-optimized resource provision for cloud applications," in *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on*, pp. 1060–1067, IEEE, 2014.

[169] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *Journal of Network and Computer Applications*, vol. 82, pp. 93–113, 2017.

[170] A. Computing *et al.*, "An architectural blueprint for autonomic computing," *IBM White Paper*, vol. 31, pp. 1–6, 2006.

[171] H. Khazaei, J. Misic, and V. B. Misic, "Performance analysis of cloud computing centers using m/g/m/m+ r queuing systems," *IEEE Transactions on parallel and distributed systems*, vol. 23, no. 5, pp. 936–943, 2012.

[172] B. Yang, F. Tan, and Y.-S. Dai, "Performance evaluation of cloud service considering fault recovery," *The Journal of Supercomputing*, vol. 65, no. 1, pp. 426–444, 2013.

[173] P. Singh, P. Gupta, and K. Jyoti, "TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud," *Cluster Computing*, nov 2018.

[174] B. Adler, "Building scalable applications in the cloud: Reference architecture & best practices, rightscale inc," 2011.

[175] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Towards an autonomic auto-scaling prediction system for cloud resource provisioning," in *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pp. 35–45, IEEE Press, 2015.

[176] J. Li, S. Su, X. Cheng, M. Song, L. Ma, and J. Wang, "Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads," *Parallel Computing*, vol. 44, pp. 1–17, 2015.

[177] J. Liu, Y. Zhang, Y. Zhou, D. Zhang, and H. Liu, "Aggressive resource provisioning for ensuring qos in virtualized environments," *IEEE transactions on cloud computing*, no. 1, pp. 1–1, 2015.

[178] M. Al-Roomi, S. Al-Ebrahim, S. Buqrais, and I. Ahmad, "Cloud computing pricing models: a survey," *International Journal of Grid and Distributed Computing*, vol. 6, no. 5, pp. 93–106, 2013.

[179] N. Joshi and S. Shah, "A comprehensive survey of services provided by prevalent cloud computing environments," in *Smart Intelligent Computing and Applications*, pp. 413–424, Springer, 2019.

[180] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE transactions on parallel and distributed systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[181] C. Qu, R. N. Calheiros, and R. Buyya, "Auto-scaling web applications in clouds: A taxonomy and survey," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 73, 2018.

[182] P. Singh, P. Gupta, and K. Jyoti, "Tasm: technocrat arima and svr model for workload prediction of web applications in cloud," *Cluster Computing*, pp. 1–15, 2018.

[183] S. S. Gill, I. Chana, M. Singh, and R. Buyya, "Radar: Self-configuring and self-healing in resource management for enhancing quality of cloud services," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 1, p. e4834, 2019.

[184] H. R. Qavami, S. Jamali, M. K. Akbari, and B. Javadi, "Dynamic resource provisioning in cloud computing: a heuristic markovian approach," in *International Conference on Cloud Computing*, pp. 102–111, Springer, 2013.

[185] M. Mohamed, M. Amziani, D. Belaïd, S. Tata, and T. Melliti, "An autonomic approach to manage elasticity of business processes in the cloud," *Future Generation Computer Systems*, vol. 50, pp. 49–61, 2015.

[186] P. D. Kaur and I. Chana, "A resource elasticity framework for qos-aware execution of cloud applications," *Future Generation Computer Systems*, vol. 37, pp. 14–25, 2014.

[187] M. Beltrán, "Automatic provisioning of multi-tier applications in cloud computing environments," *The Journal of Supercomputing*, vol. 71, no. 6, pp. 2221–2250, 2015.

[188] S. Zareian, R. Veleda, M. Litoiu, M. Shtern, H. Ghanbari, and M. Garg, "K-feed-a data-oriented approach to application performance management in cloud," in *2015 IEEE 8th International Conference on Cloud Computing*, pp. 1045–1048, IEEE, 2015.

[189] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," *ACM Computing Surveys (CSUR)*, vol. 40, no. 3, p. 7, 2008.

[190] K. RahimiZadeh, M. AnaLoui, P. Kabiri, and B. Javadi, "Performance modeling and analysis of virtualized multi-tier applications under dynamic workloads," *Journal of Network and Computer Applications*, vol. 56, pp. 166–187, 2015.

[191] M. Maurer, I. Brandic, and R. Sakellariou, "Adaptive resource configuration for cloud infrastructure management," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 472–487, 2013.

[192] M. Koehler, "An adaptive framework for utility-based optimization of scientific applications in the cloud," *Journal of Cloud Computing*, vol. 3, no. 1, p. 4, 2014.

[193] M. Ghobaei-Arani, S. Jabbehdari, and M. A. Pourmina, "An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach," *Future Generation Computer Systems*, vol. 78, pp. 191–210, 2018.

[194] S. A. Ajila and A. A. Bankole, "Cloud client prediction models using machine learning techniques," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pp. 134–142, IEEE, 2013.

[195] N. R. Herbst, N. Huber, S. Kounev, and E. Amrehn, "Self-adaptive workload classification and forecasting for proactive resource provisioning," *Concurrency and computation: practice and experience*, vol. 26, no. 12, pp. 2053–2078, 2014.

[196] S. Singh and I. Chana, "Resource provisioning and scheduling in clouds: Qos perspective," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 926–960, 2016.

[197] A. N. Toosi, C. Qu, M. D. de Assunção, and R. Buyya, "Renewable-aware geographical load balancing of web applications for sustainable data centers," *Journal of Network and Computer Applications*, vol. 83, pp. 155–168, 2017.

[198] E. Casalicchio and L. Silvestri, "Mechanisms for sla provisioning in cloud-based service providers," *Computer Networks*, vol. 57, no. 3, pp. 795–810, 2013.

[199] A. G. García, I. B. Espert, and V. H. García, "Sla-driven dynamic cloud resource management," *Future Generation Computer Systems*, vol. 31, pp. 1–11, 2014.

[200] G. Moltó, M. Caballer, and C. de Alfonso, "Automatic memory-based vertical elasticity and oversubscription on cloud platforms," *Future Generation Computer Systems*, vol. 56, pp. 1–10, 2016.

[201] M. G. Arani, M. Shamsi, *et al.*, "An extended approach for efficient data storage in cloud computing environment," *International Journal of Computer Network and Information Security*, vol. 7, no. 8, p. 30, 2015.

[202] M. S. Aslanpour and S. E. Dashti, "Sla-aware resource allocation for application service providers in the cloud," in *Web Research (ICWR), 2016 Second International Conference on*, pp. 31–42, IEEE, 2016.

[203] D. Moldovan, H.-L. Truong, and S. Dustdar, "Cost-aware scalability of applications in public clouds," in *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pp. 79–88, IEEE, 2016.

[204] A.-F. Antonescu and T. Braun, "Simulation of sla-based vm-scaling algorithms for cloud-distributed applications," *Future Generation Computer Systems*, vol. 54, pp. 260–273, 2016.

[205] M. S. Aslanpour, S. E. Dashti, M. Ghobaei-Arani, and A. A. Rahmanian, "Resource provisioning for cloud applications: a 3-d, provident and flexible approach," *The Journal of Supercomputing*, vol. 74, no. 12, pp. 6470–6501, 2018.

[206] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, and J. Chen, "A cost-aware auto-scaling approach using the workload prediction in service clouds," *Information Systems Frontiers*, vol. 16, no. 1, pp. 7–18, 2014.

LIST OF PUBLICATIONS

- **Parminder Singh,** Pooja Gupta and Kiran Jyoti. "TASM: Technocrat ARIMA and SVR Model for Workload Prediction of Web Applications in Cloud", *Cluster Computing, Springer* (Published), 2018. (Scopus, SCIE 1.6 IF)

- **Parminder Singh,** Pooja Gupta and Kiran Jyoti. "Triangulation Resource Provisioning for Web Applications in Cloud Computing: A Profit-Aware Approach", *Scalable Computing: Practice and Experience* (Accepted), 2019. (Scopus, ESCI)

- **Parminder Singh,** Pooja Gupta, Kiran Jyoti and Anand N. "Research on Auto-Scaling of Web Applications in Cloud: Survey, Trends and Future Directions", *Scalable Computing: Practice and Experience* (Accepted), 2019. (Scopus, ESCI)

- **Parminder Singh,** Pooja Gupta and Kiran Jyoti. "A Robust Auto-Scaling for Web Applications in Cloud Computing", *Cluster Computing, Springer* (Under Review), 2018. (Scopus, SCIE 1.6 IF)

- **Parminder Singh**, Gurjot Balraj Singh, and Kiran Jyoti. "A study on resource provisioning of multi-tier web applications in cloud computing", *In 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 799-802. IEEE, 2015. (Scopus)

- **Parminder Singh**, Pooja Gupta and Kiran Jyoti. "Energy Aware VM Consolidation Using Dynamic Threshold in Cloud Computing", *International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2019. (Scopus)