RESOURCE CONSUMPTION ANALYSIS OF VIRTUALISED SERVER CONSOLIDATION SYSTEM

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

BIJU R MOHAN

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

NOVEMBER 2017

DECLARATION

By the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **RESOURCE CONSUMPTION ANALYSIS OF VIRTUALISED SERVER CONSOLIDATION SYSTEM** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Information Technology** is a *bonafide report of the research work carried out by me.* The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Place: NITK, Surathkal.
Date: November 30, 2017

(IT09P03, Biju R Mohan)
Department of Information Technology

CERTIFICATE

This is to *certify* that the Research Thesis entitled **RESOURCE CONSUMPTION ANALYSIS OF VIRTUALISED SERVER CONSOLIDATION SYSTEM** submitted by **Biju R Mohan**, (Register Number: IT09P03) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of **Doctor of Philosophy**.

Prof. G. Ram Mohana Reddy
Research Guide

Prof. G. Ram Mohana Reddy
Chairman - DRPC

Acknowledgements

It is a genuine pleasure to express my deep sense of thanks and gratitude to my mentor and guide **Prof. G. Ram Mohana Reddy**, Head of the Department, Department of Information Technology, National Institute of Technology Karnataka, Surathkal, Karnataka. His dedication and keen interest above all his overwhelming attitude to help his students had been solely and mainly responsible for completing my work. His timely advice, meticulous scrutiny, scholarly advice and scientific approach have helped me to a very great extend to accomplish this task.

I owe a deep sense of gratitude to **Dr. Prakash Raghavendra**, PMTS at AMD for his keen interest, and timely suggestions that have helped me to complete my thesis. I would like to thank members of my RPAC Committee, **Prof. A. Kandasamy**, and **Prof. M. S. Bhat**, for their valuable suggestions during my research work.

I thank all the colleagues and staff profusely in the Department of Information Technology, for their help and co-operation throughout my doctoral research.

It is my privilege to thank my wife **Sunitha S**, my son **Anandapadmanabhan B**, and my daughter **Bhadra B Mohan** for their constant encouragement throughout my research period.

I am thankful to my parents **Mr. R. Rajamohan** and **Mrs. J. Indiramma** for their endless love, support and encouragement.

Biju R Mohan

Abstract

Resource consumption analysis is necessary because of continuous performance degradation of any long-running computing systems. Performance degradation is due to operating system's resource shrinkage. The most common causes of performance degradation include memory resource leakages, unreleased file descriptors, and numerical approximation errors. It is observed from literature that memory exhaustion has contributed majorly to the system failure. Resource consumption analysis is essential in a virtualized server consolidation system because Virtual Machines (VMs) use resources on demand. Another reason for selecting virtualized server consolidation system is due to the increased popularity of cloud computing. The key motivation behind this work is to help the system administrators to avoid accidental outage due to resource crunch. The key challenges in analyzing resource consumption data in server virtualized system are the volatility of the data and structural changes in the data.

First, this thesis focussed on establishing performance degradation/aging effect in virtualized server consolidation system. Then, we studied the effectiveness of ARIMA models for forecasting the resource consumption data of virtualized server consolidation system; we found the presence of heteroscedasticity in the residuals of ARIMA model. The presence of heteroscedasticity in the residuals motivated us to try heteroscedastic models like ARCH and GARCH for resource forecasting. Another hybrid model namely ARIMA-ANN is also tried for resource forecasting. By combining different models, it is possible to capture various aspects of the underlying patterns. But we have experienced a slackness of fit in all these models namely ARIMA, ARIMA-ARCH, ARIMA-GARCH, and ARIMA-ANN for the considered data. This slackness of fit is due to the presence of structural changes in the resource consumption data. Further, Regime-Switching Models like MS-GARCH and SETAR are also used to analyze the data and found that these models have reasonably fitted the considered data very well. Since there is no clear strategy for finding the order of GARCH and ARCH models, hence we tried different models and thus selected one model with least AIC, BIC, and log likelihood values for resource forecasting. An interested statistician could further investigate other mechanisms for finding the order of ARCH and GARCH models. As an extension, we would like to try these models and study the reasons for software aging in mobile platforms like Android systems in the near future.

Keywords: Resource Consumption Analysis, Software Aging, Virtualised Server Consolidation Systems, ARIMA, ARCH, GARCH, MS-GARCH, SETAR

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Resource Consumption Analysis

The importance of software reliability (Lyu 2007) and availability (Edson et al. 1996) has increased enormously due to the need of present day applications to meet many service level agreements and often software failures can cause immense economic and reputational losses. It is a well-known fact that software failures lead to more system outages than hardware failures (Gray and Siewiorek 1991, Sullivan and Chillarege 1991).

It is an observed phenomenon that the software shows increasing failure rate over time, because of a variety of reasons like resource exhaustion, the complexity of the software, nature of distribution and large size of the client groups. The progressive performance degradation of the long-running software which may lead to the system crashes or undesirable hangs and this phenomenon is referred to as software aging (Parnas 1994). The increase of failure rate accompanied by decrease of performance of a long-running software system is because of the activation and propagation of Aging-Related Bugs (ARBs).These bugs are a particular class of software faults that are visible only after a long period of the system execution. The activation of Aging-Related Bugs (ARBs) will not immediately cause a failure of the system (Avizienis et al. 2004a). The time to failure or time to exhaustion depends on the intensity of the system getting exposed to the aging-related bugs. So it may depend on the intensity of the workload of the system and also on the total running time of the system. This uncertainty in the prediction of time to failure/exhaustion is the major motivation behind this research work.

To neutralize the software aging related problems and further to enhance the availability of the system, we need a proactive measure known as software rejuvenation (Garg et al. 1997). Most of the software rejuvenation processes are initiated when the computer performance goes beyond a certain threshold. The threshold for the initiation of the rejuvenation process depends on the performance of the computer system. The need for suitable computer performance evaluation technique is a crucial problem. Most of the performance evaluation techniques depend on the analysis of resource consumption. Resource consumption implies the computing resources required for the execution of the computing system. These computing resources can be a percentage of CPU usage, power usage, physical memory usage, swap in/ swap out rate, network bandwidth, etc.

To analyze the resource consumption the resource usage data of the computing system has to be collected over a period. The workload on the system during the period of data collection must be predetermined. The resource usage data collected over a period is then analyzed using different measurement-based techniques (Section 2.2.2). Most of the researchers use time series models for analyzing such data because the resource usage collected over a period is resulting in a time series. The leading techniques are based on linear models like Auto-Regressive (AR), Moving Average (MA), Auto-Regressive Moving Average (ARMA) and Auto-Regressive Integrated Moving Average (ARIMA) models. These models are successful in places where resource consumption data do not show nonlinear dynamic patterns like asymmetry, frequency amplitude dependence, and volatility clustering. As mentioned earlier in this section that software failures will result in more system outages than the hardware failures, we briefly look into the anatomy of software failure.

1.2 Anatomy of Software Failures

Performance degradation also known as software aging is generally a result of software faults. This section briefly recollects the body of knowledge related to the taxonomy of faults and other dependability concepts.

System is a collection of interoperating elements (components); the system

boundary separates the system from its **environment**. A system can itself be a part of another bigger system. A **service failure** or **failure** of a system is a failure to meet functional requirement of a system or it can be divergence in performance of the system to meet the customer expectations. Such a divergence can be in the form of incorrect service, or no service at all. Besides this, the system will perform the service in a degraded mode in which service is sluggish or hampered; this is referred to as a partial failure. In (Avizienis et al. 2004b), the anatomy of a failure is discussed in terms of the "chain of threats", they discussed the relationship between fault, error, and failure; this chain of threats is illustrated in Figure 1.1.

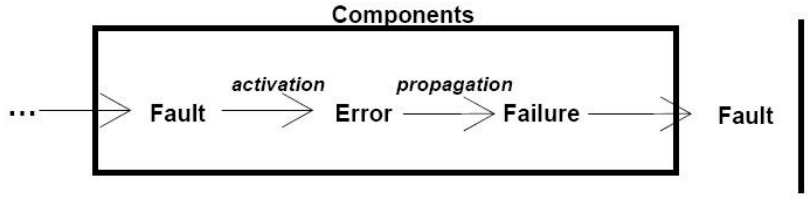

Figure 1.1: Error Propagation

An active fault can either be an internal fault, or it can be an external fault. An internal fault that was dormant previously which has been activated by a computation process or environmental conditions. This activation pattern is difficult to regenerate, so the faults or bugs remain dormant till the activation chain is initiated. Most internal faults cycle between their inactive and active states. **Error propagation** is referred to the process where an error is successively transformed into other errors. Error propagation happens when two components are connected. Imagine there is a system with two components say component 1 and component 2 are connected serially. The system will fail when a fault from component 1 propagates internally reaches the boundary of component 1 and then it enters component 2 through external propagation. It finally reaches the boundary of the system after internal propagation in component 2. This causes the failure of the system due to error propagation.

The faults could usually be classified as **Mandelbug** and **Bohrbug** (Grottke et al. 2008). A Mandelbug is hard to remove and to cause failures that are not

systematically reproducible. Generally, tester executes the part of the software where the fault is located. It does not immediately cause a failure. Rather, it produces a condition in the system that deviates from the correct internal state, this intensifies over a period of time and finally when it reaches the boundary of the system. At this stage, the user perceives this discrepancy. A Bohrbug can be easily isolated, because these bugs always reveals under a well-defined set of conditions.

1.3 Software Aging

It is observed that the failure rate of the software system increases with the increase of run-period of the system; this phenomenon is known as **Software Aging**. Failure may be an incorrect service (it might be an erroneous outcomes), halt/crash of the system, or partial failure (for example in the case of a web server a gradual increase in response time may be considered as partial failure). Hardware faults occur due to wear and tear of the equipment parts. Usually, we believe that software does not fail once the bugs are removed, and when the system is ready for final deployment, but this is not true. There are certain bugs which get activated on the longer duration of the software system's execution. These bugs are called as **aging related bugs(ARBs)**. Many **aging-related failures** of software systems are due to ARBs.

ARBs are categorized as **Mandelbugs** due to two reasons. First, because of the long delay between fault activation and the final failure happening; it is that time delay that enables the aggregation of errors. Second, the failure occurrence is due to the error propagation in the system's internal environment. The time taken for aging-related failure for a software system is random. The probability distribution of **time to aging-related failure** depends on the physical capacity of the hardware (Server, Desktop, Smartphone, an embedded system, etc.), the intensity with which the ARBs are exposed and the type of the workload. The gradual shift from correct internal state to an erroneous or a failure-probable internal state is referred as an **aging effect**. Aging effects are categorized as **volatile** and **non-volatile** effects. Volatile effects can be extinguished by reinitializing the software system. This reinitializing mechanism can either be a reboot, or it can be a partial reclama-

tion of the leaked resource. Non-volatile aging effects cannot be removed by such reinitializing mechanism. Numerical error accumulation preserved by checkpoint mechanism, file system fragmentation, database metadata fragmentation are examples of non-volatile aging effects. Examples of volatile aging effects are operating system resource leakage and physical memory fragmentation.

Software aging effects in a software system can be identified only by the execution of software system and thereby monitoring **aging indicators**. Aging indicators are system variables that can suggest whether the system is healthy or not. Aging indicators can be monitored at several levels, such as operating system, at the application level, middleware, virtual machine (VM), and VM monitor (VMM). Examples of aging indicators at system level are free physical memory, the % of CPU usage, file table size, used swap space, and network bandwidth, etc. Java VM heap size and response time are examples of aging indicators at application specific level.

1.3.1 Software Rejuvenation

As we know, it is not possible to fix aging-related bugs, because of two constraints. The first constraint is that these bugs are in proprietary third-party code which developers do not have the source code and the second constraint is the complexity of nowadays software applications. Software rejuvenation (Cotroneo et al. 2014) is a fault-tolerance mechanism to mitigate performance degradation and other aging-related failures. Research at the AT&T Bell Laboratories (Huang et al. 1995) identified this proactive mechanism, and since 1990 it has been used as a cost-effective solution in several fault-tolerant software to fight against software aging.

Huang et al. 1995 defined software rejuvenation as the "pre-emptive rollback of continuously running applications to prevent failures in the future". The software system will be inaccessible during rejuvenation, there may be a small downtime associated with rejuvenation, and this may lead to a small loss of revenue due to the loss of business. However, the costs due to downtime can be reduced by scheduling rejuvenation during the idle time of the software system. This is practically an excellent proposition because an unexpected outage may incur more revenue loss.

Hence, one of the critical issues in the software aging and rejuvenation area is to find the time to rejuvenate (rejuvenation schedule); this will improve availability and minimize revenue loss due to the outage of software system during busy time.

Huang et al. 1995 discussed software rejuvenation (Figure 1.2) by a simple state transition model based on continuous-time Markov chains. On initiation, the system is in highly stable state S_0, in which the probability of aging-related failures is negligible. At a rate, r_2 the system slowly moves into a state S_P which is failure probable state (where some operating system resources have been leaked and are near to being exhausted). From this state, the system can fail at the rate λ and reach a failed state S_F and has a repair rate of r_1. If the system undergoes rejuvenation, it will go from the failure probable state S_P to S_R rejuvenated state at a rate of r_4, and then to the stable state S_0 at a rate of r_3. The downtime and revenue loss for the rejuvenation path ($S_0 \rightarrow S_P \rightarrow S_R \rightarrow S_0$) is much lesser than the failure path ($S_0 \rightarrow S_P \rightarrow S_F \rightarrow S_0$).

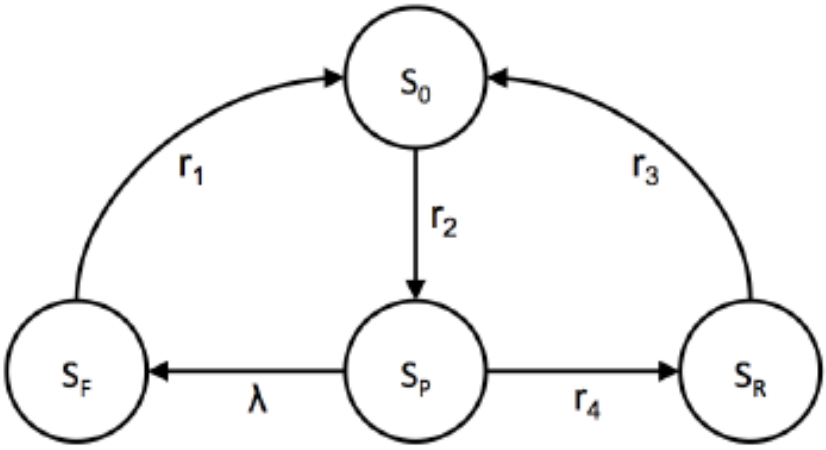

Figure 1.2: State Transition Model of a System with Rejuvenation

1.4 Virtualized Server Consolidation System

Server consolidation allows efficient usage of server resources; this is done by reducing the number of servers in an organization. Server consolidation is an energy-saving solution practised to avoid server sprawl. Server sprawl is a situation where several underutilized servers are deployed and it consumes more space and resources than

justified by their workload. Some advantages of server consolidation are as follows.

- Lower power and cooling costs
- Lesser hardware costs
- Flexibility to migrate workloads between physical servers
- Reduced workforce
- Better software pricing - Fewer licenses may be required
- Minimum physical space
- Easier Disaster Recovery - Easy to maintain, backup and restore with fewer systems.

Resource usage/consumption analysis is essential for a server virtualized consolidation system because Virtual Machines (VMs) use resources on demand. The reason for selecting server virtualized systems is due to the increased popularity of cloud computing, and the resource allocation in such a system is dynamic in nature. The dynamic nature of allocating resources in server virtualized system makes resource consumption analysis more challenging. Virtualization is the key aspect of virtualized server consolidation system, which is discussed in the following section.

1.4.1 Types of Virtualization

Virtualisation is classified as, 1. Server Virtualisation 2. Desktop Virtualisation 3. Storage Virtualisation 4. Network Virtualisation.

- **Server Virtualization** In server virtualization, the system administrator uses a virtualization software to segregate one physical server into multiple isolated virtual environments. Server virtualization is based on the software architecture called virtual machine, it abstracts actual hardware using a software layer, and the operating system is installed on top of this layer of software abstraction.

- **Desktop Virtualisation** Desktop Virtualisation is defined as a virtualization technology that is used to separate a computer desktop environment from the physical machine. The virtualised desktop is stored on a remote server and it uses a client-server software architecture. It virtualizes desktop computers, and these virtual desktop environments are delivered to users through the network.

- **Storage Virtualization** Storage virtualization creates a layer of abstraction between the operating system and the physical disks used for data storage. In Storage virtualization, a pool of multiple network storage devices forms a single storage device, which could be managed from a central console.

- **Network Virtualisation** Network virtualization combines network resources and network functionality into a single unit called a virtual network. We can administer that with software.

1.4.2 Types of Server Virtualization

There are two types of server virtualization, namely (1) Full virtualization or Bare-Metal Virtualization and (2) Para-virtualization or Hosted Virtualization. A hypervisor or virtual machine monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines. There are two types of Hypervisors or Virtual Machine Monitors (VMMs) namely, 1) Type 1 VMM Bare-Metal VMM and 2) Type 2 VMM or Hosted VMM. In Full Virtualization, the Virtual Machine Monitor has direct access to hardware resources resulting in better scalability, stability, and performance. Unlike the full virtualization VMM or Type 1 VMM, a paravirtualization VMM or Type 2 VMM requires an operating system, paravirtualization VMMs are basically like applications that are installed on a guest operating system. The full virtualization is popular among different types of server virtualization because it offers better isolation and security for VMs, and simplifies the migration and portability. Another advantage of full virtualization is that it avoids the extra layer of abstraction as in the case of para-virtualisation. Most of the experiments in this

thesis are done using Type 1 VMM. The Figure 1.3 shows the software architecture of hosted and bare-metal hypervisors.

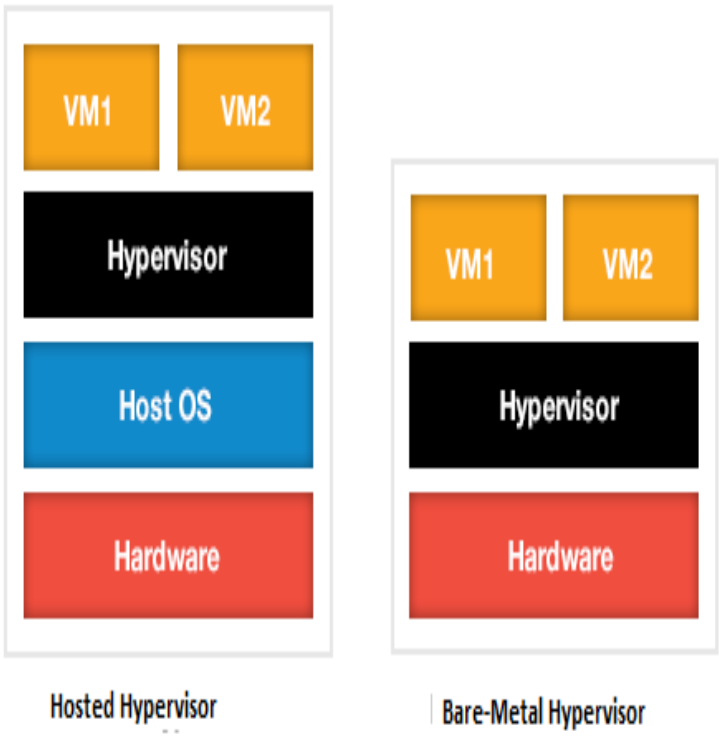

Figure 1.3: Different Types of Hypervisors

1.5 Motivation

Resource consumption analysis is essential in most of the computing environments where performance is one of key non-functional software requirements. In the section, we are explaining some of the important reasons why resource consumption analysis is vital in a virtualized server consolidation system.

- **Helps system administrator to avoid an accidental server outage.**

  An accidental outage of long running servers occurs when the operating system resources are exhausted due to aging-related bugs. From (Matias et al. 2010, Jr. et al. 2010, Zhao et al. 2011), it is found that memory exhaus-

tion contributes majorly to the system failure due to the resource shortage. The accidental server outage brings huge revenue loss as well as goodwill loss hence an accidental server outage is a nightmare for system administrators. To avoid an accidental outage system administrator has to monitor, forecast and analyze the operating system resources on a day-to-day basis.

- **To meet certain non-functional software requirements.**

  In cloud computing environments, the resource consumption analysis is essential to meet certain non-functional software requirements like the acceptable levels response time, minimum mean time to repair, high availability, etc.. The response-time of the aged server increases with the propagation of aging-related bugs. This could be avoided if the memory leakage is tracked in advance. Similarly, the mean time to repair could be minimized, and the availability could be increased by detecting the performance degradation in earlier stages and thereby applying appropriate rejuvenation mechanism. Most of the performance degradation methods rely on resource consumption analysis.

- **To estimate the time for live migration of Virtual Machines.**

  To help the system administrator in estimating the time for live migration of virtual machines (VMs), the resource consumption analysis is needed. The live migration of VMs happens when the resource usage exceeds the physical limit of the server capacity or when the performance degrades below the service level agreement(SLA). Resource consumption analysis is essential for finding the resource usage pattern and thus identifying the performance degradation.

- **Helps server capacity planning**

  Inadequate server resource planning will lead to increased expenditure for server hardware. Resource consumption analysis helps us to quickly see what is causing server resource contention. It will track how much CPU, memory and storage hardware is being consumed. It helps system administrator to right-size resource allocation on physical and virtual servers, and thus optimizes the performance.

1.6 Issues and Challenges

The major issues and challenges of resource consumption analysis are as follows:

- **Non-availability of Data-centre Resource usage data**

  The first and foremost challenge faced is the availability of suitable dataset where we could do the analysis. Most of the data centers are not ready to share their resource usage data log. We need to overcome the situation by setting up an experimental environment where we could be able to collect necessary data. Another issue with such experimental setup is the difficulty of simulating the heterogeneity in the transactional behavior of the application server which leads to dynamicity in the resource usage pattern.

- **Long Experimentation Time**

  The second issue is the long experimentation time. The time required for the propagation of the aging-related bugs depends on the intensity with which the bug is activated. The intensity with which the bug is activated cannot be controlled by experiments as the bugs are transient in nature. The aging effect could impact the resource usage pattern of the system, so tests require long experimentation time. Based on the data analysis techniques applied at least a few number of failures of the system under test (in this case virtualized server consolidation system) need to be noticed. The problem worsens when the system under test is composed of highly reliable components, such as those developed for telecommunications, military, safety-critical, and long-life systems. In this case, the system under test falls in the category of long-life systems.

- **Existence of Heteroscedasticity**

  If the subpopulations of a time series have different variabilities from one another, then the time series is said to be heteroscedastic in nature. The presence of heteroscedasticity in the resource usage dataset is a major concern when

applying any regression models for analyzing the time series. Heteroscedasticity in the dataset can nullify statistical tests of significance that assumes that the modeling errors are uncorrelated and uniform. Even in the presence of heteroscedasticity, the ordinary least squares estimator is still unbiased, but it is inefficient because the actual variance and covariance are underestimated.

- **Presence of structural breaks in the time series data**

  While using a regression model involving time series data, a structural change may occur in the relationship between the regressand and the regressors. Structural change means that the values of the parameters of the regression model do not remain the same through the entire time period of the dataset. The structural change in the resource usage dataset occurs when the system under test started aging. Due to aging, we could see a massive shift in the mean and variance of the data when we consider different subpopulations of the dataset. Finding the number of structural change is a resource challenge in time series analysis. Here we can find that the existence of structural variations in the dataset has tremendously affected the goodness of fit of ordinary linear and nonlinear models and hence we used regime switching models to acclimate such changes.

1.7 The Complete Research Framework

The complete research farmework described in the thesis is shown in Figure 1.4 and it consists of the following steps.

1. Aging Trend Detection
2. Resource consumption analysis using Auto-Regressive Integrated Moving Average (ARIMA) Model
3. Resource usage analysis with Auto-Regressive Conditional Heteroskedasticity (ARCH), Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH), and Artificial Neural Network (ANN) Models

4. To study the appropriateness of regime switching models for resource consumption analysis

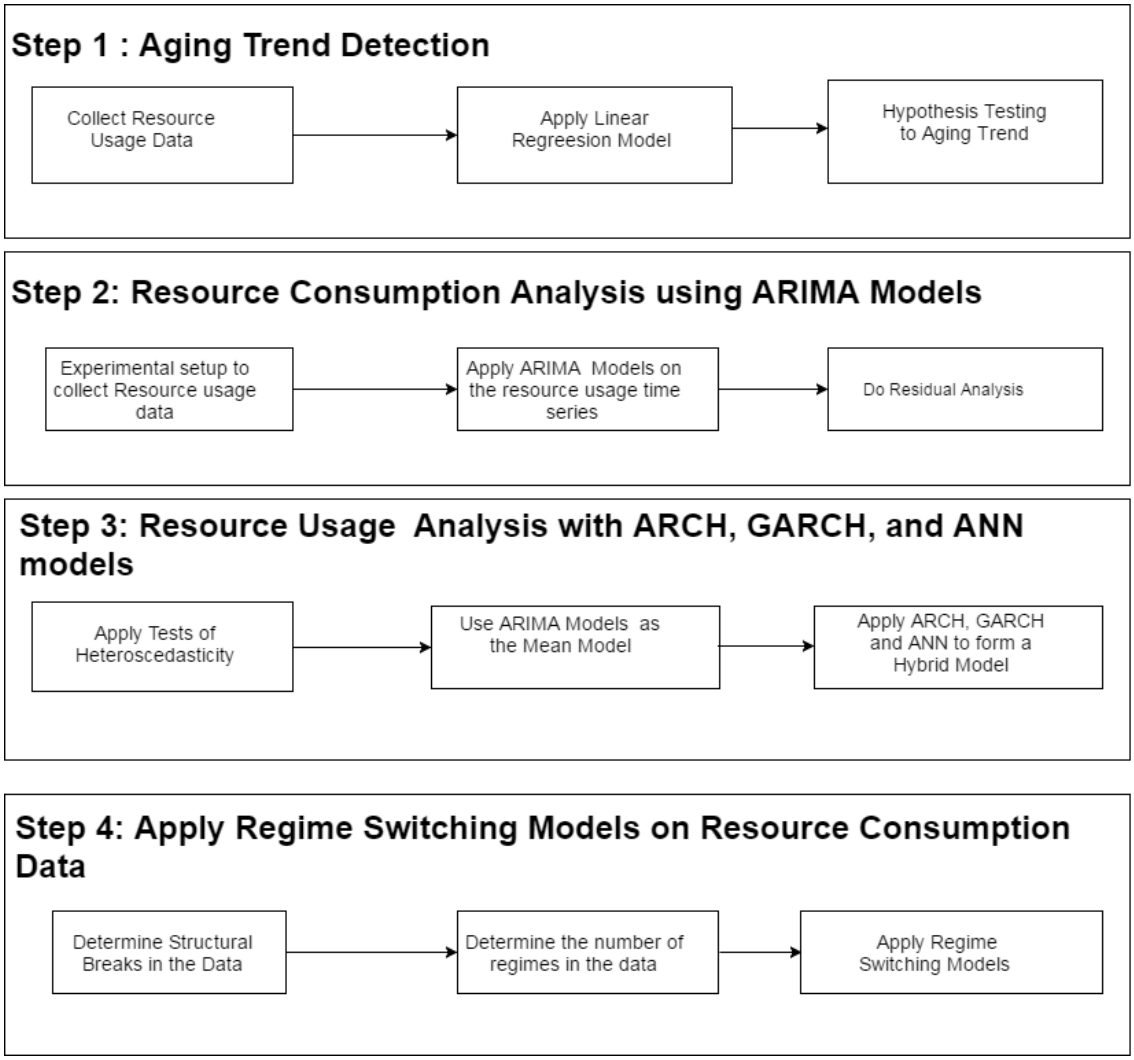

Figure 1.4: The Complete Research Framework

- **Aging Trend Detection:** The main focus in this step is to detect the software aging effect in the virtualized server consolidation system. The thesis establishes the aging effect by demonstrating that the average response time of web servers loaded in the Virtual Machines of the server consolidation system decreases over a period for a constant workload and it is observed that free physical memory decreases during this period. The Linear regression model is considered to detect the aging effect.

- **Resource Consumption Analysis Using ARIMA Model:** In this step, resource consumption data obtained from an experimental setup is analyzed using an ARIMA model. Here we check the adequacy of the ARIMA model for forecasting the future resource usage.

- **Resource Usage Analysis with ARCH, GARCH and ANN Models:** This step reviews the effectiveness of linear time series models for the resource consumption data. And it examines whether volatility is present in the resource usage data or not. Further, this step checks whether Gauss-Markov assumptions about homoscedasticity holds good for the ordinary least square estimators of linear time series models or not. This step applies a combination of models, namely, Auto-Regressive Integrated Moving Average - Artificial Neural Network (ARIMA-ANN), Auto-Regressive Integrated Moving Average - Generalized Auto-Regressive Conditional Heteroskedastic (ARIMA-GARCH), and Auto-Regressive Integrated Moving Average - Auto-Regressive Conditional Heteroskedastic (ARIMA-ARCH).

- **Regime Switching Models on Resource Consumption Data:** Resource-consumption time series data exhibits structural breaks in their behavior. A structural break is an unexpected shift in a time series and this can lead to forecasting errors and also causes unreliability of the model in general. The resource consumption data exhibits structural break mostly when the system is aged. This step analyses the appropriateness of regime switching models for resource usage analysis.

1.8 Summary of Research Contributions

The following are the research contributions of this thesis:

- The thesis extensively reviewed the methods available for aging trend detection; the study found that there is no work done in the area of server consolidation system. In this work, we established aging trend in the server consolidation system, and we proved that the response time of web servers in the

server consolidation system increases for the same workload. We proved that the power consumption increases due to the system performance degradation for the same workload.

- This thesis studied the effectiveness of ARIMA models for forecasting the resource consumption data of virtualized server consolidation system and thereby finding the presence of heteroscedasticity in the residuals of ARIMA fit. We found that ARIMA models are not suitable for forecasting resource consumption data due to the presence of non-linear components like volatility clustering and structural breaks.

- The thesis then verified the appropriateness of heteroscedastic models like Auto-Regressive Conditional Heteroskedasticity (ARCH) and Generalized Auto-Regressive Conditional Heteroskedasticity (GARCH) models. These models are used to analyze the resource consumption data. We also tried a hybrid ARIMA-ANN (Auto-Regressive Integrated Moving Average - Artificial Neural Network) model.

- The thesis highlighted the presence of structural breaks at the start and end of entire time series data from the workload initiation till we reach the failure of virtualized server consolidation system. Further, we analyzed the suitability of two regime switching models namely Markov-Switching Generalized Auto-Regressive Conditional Heteroskedasticity (MSGARCH), and Self-Exciting Threshold Auto-Regressive (SETAR). We found that MS-GARCH and SETAR models have reasonably fitted the resource consumption data.

1.9 Organisation of the Thesis

The remainder of the thesis is organized as follows. Chapter 2 presents an overview of the recent works in the area of resource consumption analysis and software aging analysis. It studies merits and demerits of measurement based and model based methods in detecting the aging effect. Chapter 3 explains the experimental setup for collecting the data, and it applies linear regression model on the response time of the

webservers loaded in the VMs. Then it applies the statistical hypothesis testing for determining the aging trend. Further in Chapter 3, we discuss the appropriateness of data analysis using ARIMA model.

Chapter 4 checks whether Gauss-Markov assumptions about homoscedasticity holds good for the ordinary least square estimators of linear time series models or not. Further, it analyses two heteroscedastic models namely ARCH and GARCH models. It examines the appropriateness of a hybrid ARIMA-ANN model in analyzing the resource consumption data. Chapter 5 discusses the structural breaks in the data and analyses the suitability of two regime switching models namely MS-GARCH and SETAR for analyzing the resource consumption data. Chapter 6 concludes the research work and highlights the possible future research directions.

1.10 Summary

This chapter started with an introduction to resource consumption analysis. It scrutinizes the internal working of the software failure. Further, this chapter introduced the error propagation mechanism and acquainted different bugs and it familiarized software aging and rejuvenation. This chapter then discussed issues and challenges, motivation, comprehensive research framework, research contributions and finally the organisation of this thesis. Next chapter discusses the existing works in software aging and rejuvenation.

Chapter 2

Literature Review

2.1 Software Aging and Rejuvenation

Software aging is a situation where complex, long-running software systems exhibit increasing failure rate. Before failure the software systems usually exhibit performance degradation, this is due to the error propagation, system resource leakage especially memory leakage (Grottke et al. 2008,Huang et al. 1995). But this is not a new phenomenon as researchers were studying about this since 1960. Nowadays software systems are becoming more complex; we would expect the number of software systems affected by software aging also increase. It is mentioned in (Grottke et al. 2008), that software aging is due to Mandelbugs which is noticeable when the error propagates. This is evident as unreleased file locks, memory leakage and numerical error accumulation, which degrade the performance of the system and eventually cause the system failure. Rejuvenation is a proactive measure to bring back the system from a failure probable state to a healthy state.

Cotroneo et al. 2014 categorized the software aging and rejuvenation studies into four based four dimensions, namely, 1) Type of Analysis, 2)Type of system, 3)Resource Consumption Analysis, and 4)Rejuvenation techniques. Figure 2.1 gives pictorial representation of the categorization of research studies in software aging and rejuvenation.

- **Type of Analysis:** There are three different types of analysis usually found in the literature 1) Measurement based Methods 2) Model-based Methods and 3) Hybrid Methods. Here most of the existing works deal with novel techniques

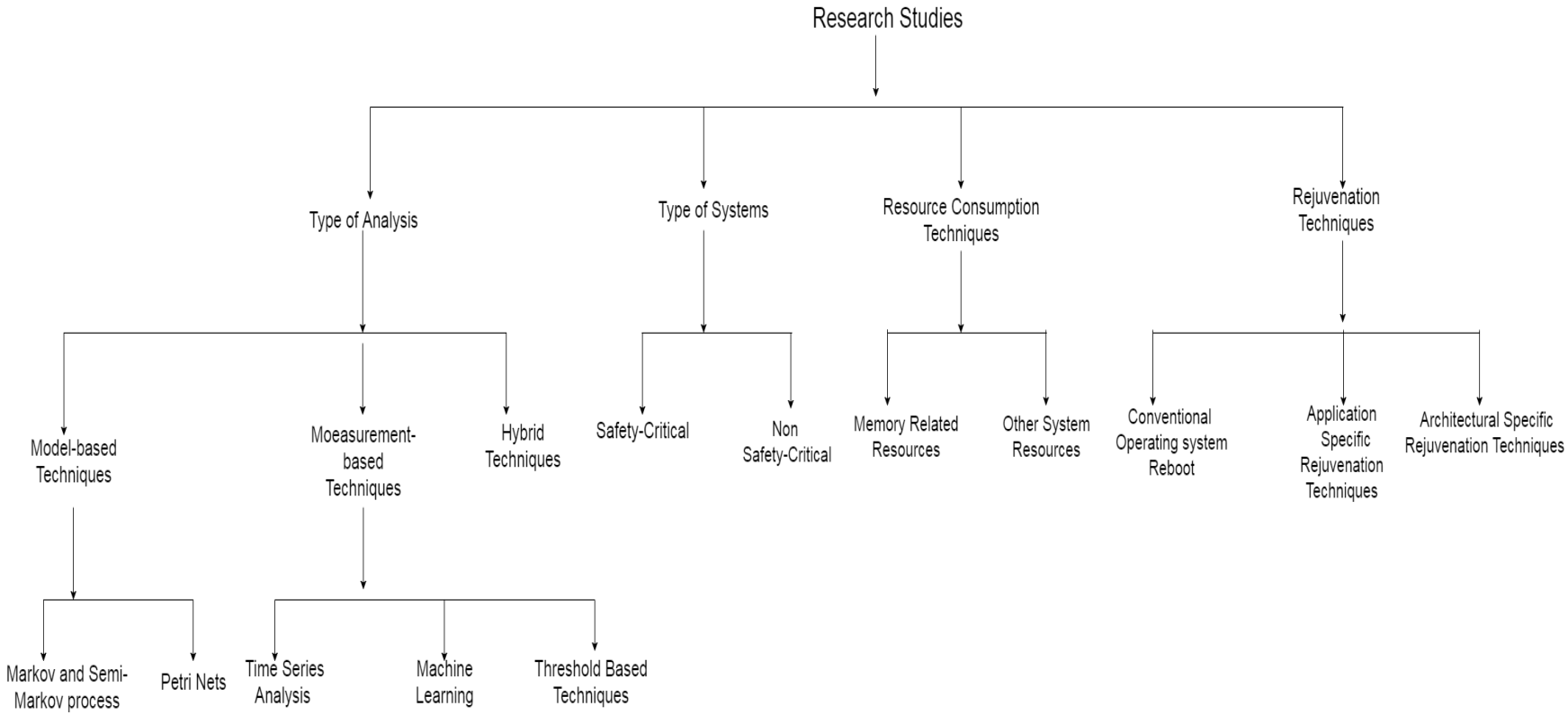

Figure 2.1: Categorization of Research Studies in Software Aging and Rejuvenation.

in terms of finding the software aging trend detection, resource consumption analysis, finding the time to rejuvenate, new rejuvenation techniques and failure data analysis.

- **Type of System:** Generally, the types of the system can be classified as safety critical and non-safety critical systems based on the domain and criticality of the system. Some systems are critical from the safety perspective and the failure of such systems may cause human fatalities. The software systems used in war, medicine, air traffic control and in some industrial control could come in this category. Non-safety critical systems are systems used in business, and mission-critical application where its accidental stoppage may cause huge revenue and goodwill loss.

- **Resource Consumption Techniques:** The gradual shift of a system from a healthy state to a failure-probable state is reffered to as an aging effect. System resource usage/consumption analysis is essential in identifying the aging effect of the system. The research work in software and rejuvenation may be classified based on different types of resource consumption analysis; Memory consumption, Network resource usage, application specific resource usage, CPU usage, Concurrency-related resource, and file system related resources are some of the system resources commonly used for studying the aging phenomenon.

- **Rejuvenation Techniques:** The last dimension in which the software aging and rejuvenation studies are classified is based on the rejuvenation techniques. Rejuvenation is a procedure (a set of steps) followed by software engineers to revitalize the system from a failure-probable state to healthy state and thereby avoiding an unexpected outage. There are many application-specific and architectural specific rejuvenation techniques practiced nowadays. Application restart, Operating System reboot, partial restart of certain components, VM restart, migration of VM, Java Virtual Machine (JVM) restart, VMM restart, etc. are a few techniques.

In this thesis, we will focus mainly on type of analysis and resource consumption techniques, as this is more relevant to this work. Type of system used for the aging studies will be considered throughout the literature survey. We will not review different rejuvenation techniques as this is not relevant for this thesis work.

2.2 Type of Analysis

The section examines the type of analysis used in the existing works. The techniques used by several researchers across the world to study software aging and rejuvenation could be broadly classified into three, namely 1) Model-based studies 2) Measurement-based studies and 3) Hybrid-studies.

- **Model-based Studies:** These studies are based on abstract models, which usually make assumptions about the characteristics of the system. These models assume probability distributions of the system variables. These models could be applied to any system; it is portable because these model-based techniques never capture the specific details of the system under test. In some cases, model-based techniques rely on the real data collected from measurement-based techniques.

- **Mesurement-based Studies:** A lot of work is done in the area of measured-based techniques. These methods are based on the performance analysis, where the real system is used for the study. Measurements of the system variables, i.e., aging indicators are measured directly from the actual system. Assumptions are usually not made in this technique as it collects more specific information about the system under study.

- **Hybrid Studies:** Considerable research is done to combine the right aspects of both model-based and measurement-based approaches; these works are referred to as hybrid. In spite of the practical importance of hybrid approaches, these techniques are not popular among the researchers.

2.2.1 Model-based Studies

Model-based studies are usually based on the stochastic processes; the most commonly used methods in model-based studies are Markov and Semi-Markov processes. Petri Nets are also one of the popular models among the model-based studies. These model-based studies can be classified into two, namely 1) Methods based on Markov and Semi-Markov processes and 2) Methods based on Petri Nets.

Techniques based on Markov and Semi-Markov Processes

Huang et al. 1995 employed an analytical model to study about the revenue loss associated with the downtime of an aged system. But they did not specify the type of the aging system. They analytically proved that rejuvenation of a system which is in failure probable state could reduce the revenue loss due to an unexpected outage. Further, they used a probabilistic State Transition model based on Continuous-Time Markov Chain.

Garg et al. 1998a analytically demonstrated the necessity of preventive maintenance to fight software aging. They used non-homogeneous Continuous-Time Markov Chain model to represent transaction based software systems in which preventive maintenance is available. They applied two preventive maintenance policies, namely; one purely based on the time and the other based on the time and the workload. They considered some numerical samples to show that the policy based on the time and the workload have reduced optimum probability loss. They assumed that the incoming transactions were lost when the system fails but it is not true in many transactions based software systems.

Bao et al. 2005 used Continuous-Time Markov Chain to model an operating software system with resource leaks and further they used a three-state Semi-Markov process to model rejuvenation. They established a theoretical connection between operating system resource leakage and software aging. They assumed that the reactive recovery time of the system is roughly 10-12 times more than the recovery time of a system with rejuvenation but it has no measurement based validations.

Myint and Thein 2010 considered a Markov chain based model to map software rejuvenation process of a virtualized server system. They modeled multiple physical servers which host multiple virtual machines (VMs) and evaluated the performance through numerical analysis and also by using a simulation tool. They showed that software rejuvenation and virtualization could enhance the availability of the services. But they did not mention the specific virtualization technique used. Further, their numerical analysis has no experimental validations.

Koutras and Platis 2011 studied the effect of partial and full rejuvenation on system performance and dependability. The system performance is gauged through downtime, availability and rejuvenation cost. They used Continuous Time Markov Chain to model the software system, but they did not consider the real life system. Table 2.1 summarizes the merits and demerits of methods based on Markov and Semi-Markov Processes.

Techniques based on Petri Nets

Wang et al. 2007 designed three rejuvenation policies to improve the performance of cluster under the varying workload. They used a Deterministic and Stochastic Petri nets (DSPN), where the stochastic process is a Markov regenerative process (MRGP). Among the three rejuvenation policies, namely standard rejuvenation, delayed rejuvenation and mixed rejuvenation, the mixed rejuvenation policy achieves the maximum performability. The performability is analyzed based on blocking probability, availability, and throughput of the system, but they did not validate the results with the real data. Vaidyanathan et al. 2001 employed Stochastic Reward Nets (SRNs) to model and analyze software rejuvenation policy in cluster systems. They developed time-based and prediction-based rejuvenation policies and compared their performance based on system availability and cost. They did not consider the hardware failures but they assumed ideal values for common mode failure.

Salfner and Wolter 2010 investigated the impact of 3 time-based rejuvenation policies on service availability. They used Deterministic and Stochastic Petri Nets (DSPN) to estimate the metrics needed to compute service availability. The high

Table 2.1: Merits and Demerits of Methods based on Markov and Semi-Markov Processes

Works	Model Used	Type of System	Merits	Demerits
Huang et al. 1995	State Transition model based on Continuous Time Markov Chain.	No specific system.	Analytical proof for rejuvenation in reducing the revenue loss due to an unexpected outage.	No directions on how to apply resource consumption analysis and identify aging effect.
Garg et al. 1998a	Continuous-Time Markov Chain.	Represented transaction based software systems.	Demonstrated the necessity of preventive maintenance to fight software aging.	The assumption of loosing incoming transactions when the system fails is not true in many transactions based software system.
Bao et al. 2005	Continuous-Time Markov Chain and Semi-Markov process.	Represented Operating systems.	Established a theoretical connection between operating system resource leakage and software aging phenomenon.	The assumption of the reactive recovery time of the system without rejuvenation is roughly 10-12 times more than the recovery time of a system with rejuvenation has no measurement based validations.
Myint and Thein 2010	Continuous-Time Markov Chain.	Represented physical servers which can host multiple virtual machines (VMs).	Evaluated the performance through numerical analysis and simulation tool, showed that software rejuvenation and virtualization could enhance the availability of the services	The numerical analysis does not have experimental validations.
Koutras and Platis 2011	Continuous-Time Markov Chain.	No specific system.	Studied the effect of partial and full rejuvenation on system performance and dependability.	Numerical results presented were not from the real life system.

level of abstraction makes the model portable, but the specific details of the system cannot be captured using such models. Andrade et al. 2011 developed a method to compare the rejuvenation policies of a server system using Stochastic Reward Nets and SysML. They developed a component-based availability modeling framework named Candy. But this method needs to be validated in a wider context to know its applicability. Table 2.2 summarizes the merits and demerits of methods based on Petri Nets.

Model-based studies usually rely on abstract models and with certain assumptions about the underlying system; sometimes there may not be any particular system for consideration by the researcher when they go for a very high level of abstraction. Specificities of the system are never captured by the researchers in this type of analysis. One distinct advantage of model-based studies is that it could be applied to all systems or even to systems in the pre-production stages (after requirement analysis) as these methods do not require real system. The scalability of the results of the model-based analysis is medium and the accuracy of results of the model-based analysis is less when compared to the measurement-based studies.

2.2.2 Measurement-based Studies

Measurement-based studies are most popular techniques among the software aging and rejuvenation analysis; the primary reason is that the results obtained from these studies are more accurate than other types of analysis. The main reasons for this higher accuracy are: 1) Real system is used in the measurement-based analysis, and 2) Specificities of the system is incorporated while designing an experiment to collect the data from the system. The research using measurement-based analysis could be classified based on the tools used for analysis. The measurement-based studies are mainly classified into three techniques, namely: 1) Time-series analysis based, 2) Machine Learning based, and 3) Threshold based methods.

Table 2.2: Merits and Demerits of Methods based on Petri Nets

Works	Model Used	Type of System	Merits	Demerits
Wang et al. 2007	Deterministic and Stochastic Petri nets (DSPN)	Represented cluster based system.	Designed and compared three rejuvenation policies to improve the performance of cluster under the varying workload.	Results were not validated with the real data.
Vaidyanathan et al. 2001	Stochastic Reward Nets (SRNs)	Cluster systems.	Developed time based and prediction based rejuvenation policies and compared their performance.	Not considered hardware failures and assumed ideal values for common mode failure.
Salfner and Wolter 2010	Deterministic and Stochastic Petri Nets (DSPN)	No specific system.	Investigated the impact of 3 time-based rejuvenation policies on service availability.	Specific details of the system cannot be captured.
Andrade et al. 2011	Stochastic Reward Nets and SysML	No specific system.	Developed a method to compare the rejuvenation policies of a server system.	The method is not tested in wider and specific context.

Techniques based on Time-series Analysis

In the time series analysis based techniques, the software aging indicators of the system under study are directly measured over a period; the measurement thus obtained will result in a time series. The studies using time series analysis usually falls in the category of aging trend analysis, resource consumption, forecasting, and seasonality pattern prediction.

Garg et al. 1998b designed a method for detection and estimation of aging in UNIX operation system. They used a Simple Network Management Protocol agent to collect data from remote UNIX servers. The aging indicators include used swap space, file table size, free available memory, disk I/O activity and network-related resources. They studied periodicity, linear dependence, and trend. They applied seasonal Kendall test to each of these time series to detect the presence of aging patterns. They used linear regression methods for estimating the true slope of the trend. But they did not carry out the residual analysis to know the appropriateness of fit.

Li et al. 2002, Grottke et al. 2006 designed a method based on Auto-Regressive Moving Average (ARMA) model for estimating resource exhaustion in a Web server. They collected response time, free physical memory and used swap space of an Apache web server and used the Man-Kendall test to determine the trend of the parameters. A non-parametric procedure developed by Sen 1968 is used to estimate the slopes of the variables. The results were compared with previous measurement-based technique and found more efficient. But their developed model is specific to Apache server, and the presence of volatility in the data is not analyzed.

Cotroneo et al. 2010 studied software aging in a Linux Kernel. They developed a kernel tracing tool to monitor system variables like page misses, page cache flushes, cache insertions, interrupt requests, etc. They used Mann-Kendall test for trend detection. Multiple Linear Regression models were used to study the relation between the aging and system variables. To avoid multicollinearity, they used principal component analysis (PCA). The analysis of time to rejuvenation was missing in the study and the reasons for the aging trend in Linux kernel was not analyzed.

Cotroneo et al. 2013 analyzed software aging in a Java Virtual Machine (JVM). They revealed some aging effects like throughput loss and Memory leakage in both Just-in-Time (JIT) compiler and Garbage collector. They used hypothesis testing to establish aging trend and used multiple linear regression to estimate the aging trend. They applied hypothesis testing on principal components and multiple regression for the assessment of the aging-workload relationship. The goodness of fit of these models was not discussed in their work.

Magalhaes and Silva 2010 developed a framework to monitor system parameters, which is used to determine the correlation between the application response time and workload. They used ARIMA (Autoregressive Integrated Moving Average) and Holt-Winters (Triple Exponential Smoothing) models to identify performance deviation on the web-based and component-based applications. The goodness of fit of models was not analyzed. The effectiveness of their method is specific to the scenario, so it has to be tested on other applications to know its efficiency on the different platforms. Table 2.3 summarizes the merits and demerits of techniques based on Time-series analysis.

Techniques based on Machine Learning Algorithms

Machine learning algorithms are one of the favorite techniques used by several researchers across the world in the area of software aging/ rejuvenation and some key works are discussed as follows.

Cassidy et al. 2002 developed a mechanism for analyzing software aging in online transaction processing (OLTP) servers. They identified that latch contention is the primary cause of performance degradation in DBMS server. They investigated the feasibility of applying a pattern recognition algorithm to detect software faults. They used a proprietary software (SmartSignal) based on Multivariate State Estimation Technique to predict possible failures. They suggested a manual rejuvenation by flushing DBMS buffers. Alonso et al. 2011 studied software anomalies in an E-Commerce environment using machine learning algorithms like Decision Tree, Naive Bayes, and Support Vector Machines for classification. They used Apache Tomcat

Table 2.3: Merits and Demerits of Methods based on Time-series Analysis

Works	Model Used	Type of System	Merits	Demerits
Garg et al. 1998b	Seasonal Kendall test and Linear Regression Models	UNIX operation system.	Designed a method for detection and estimation of aging in UNIX operation system.	Residual analysis is not done.
Li et al. 2002,Grottke et al. 2006	Auto-Regressive Moving Average (ARMA) and Man-Kendall test	Apache web server.	Designed a method for estimating resource exhaustion in a Web server.	The model developed is specific to Apache server, and the presence of volatility in the data is not analyzed
Cotroneo et al. 2010	Multiple Linear Regression, Principal Component Analysis (PCA) and Man-Kendall test	Linux Operating System.	Designed a method for software aging analysis in a Linux Kernel.	The analysis of time to rejuvenation is not done.
Cotroneo et al. 2013	Multiple Linear Regression, Principal Component Analysis (PCA) and Hypothesis Testing	Java Virtual Machine (JVM).	Found some aging effects like throughput loss and Memory leakage in both Just-in-Time (JIT) compiler and Garbage collector.	Goodness of fit of these models is not discussed in this work.
Magalhaes and Silva 2010	ARIMA (Autoregressive Integrated Moving Average) and Holt-Winters (Triple Exponential Smoothing).	Web-based and component-based applications.	Determined the correlation between the application response time and workload.	Goodness of fit of these models is not discussed in this work.

as a web server and MySQL as a Database server. They simulated a multi-tier e-commerce on-line bookstore, following the standard configuration of TPC-W (TPC-W is a web server and database performance benchmark, proposed by Transaction Processing Performance Council) benchmark. The effectiveness of these methods were not compared with time series methods.

Magalhaes and Silva 2010 developed a framework to monitor system parameters and this framework is used to determine the correlation between the application response time and workload. They forecasted performance failures, caused by software aging on the web-based and component-based applications. The forecasting is done using a hybrid model which is a combination of machine learning (ML) classification algorithms and time-series analysis. They used the Naive Bayes, Decision Trees, and Artificial Neural Network for classifying the forecasted aging indicators. They used ARIMA and Holt-Winters models to forecast the aging indicators. The goodness of fit of these models was not discussed in this work. Yan and Guo 2016 developed a method based on machine learning algorithms like decision tree algorithms, artificial neural network algorithms (ANN), and support vector machine algorithms (SVM) to predict the software aging. They applied this method to an IIS web server, analysed the experimental results and found that their method can predict software aging reasonably well. The analysis of time to rejuvenation was missing in their study. Table 2.4 summarizes the merits and demerits of techniques based on Machine Learning algorithms.

Threshold-based Techniques

In this approach, basically the system resources which are the cause of aging effect are studied. The studies define a threshold for the system resources, and rejuvenation is triggered when the monitored resources exceed such thresholds. The major difficulty is in identifying the relevant system resource and the right thresholds; this will prevent the system from unexpected failure. Relevant works using threshold-based techniques are discussed in this section. Silva et al. 2009 developed a method for software rejuvenation for application servers; it is based on automated

Table 2.4: Merits and Demerits of Methods based on Machine Learning

Works	Model Used	Type of System	Merits	Demerits
Cassidy et al. 2002	Multivariate State Estimation Technique	Online transaction processing (OLTP) servers.	Identified that latch contention is the primary cause of performance degradation in DBMS server.	Only manual rejuvenation by flushing DBMS buffers.
Alonso et al. 2011	Decision Tree, Naive Bayes, and Support Vector Machines for classification.	Simulated a multi-tier E-commerce on-line bookstore.	Studied software anomalies in an E-commerce environment.	The effectiveness of these methods are not compared with time series methods.
Magalhaes and Silva 2010	Naive Bayes, Decision Trees, and Artificial Neural Network for classification. Autoregressive Integrated Moving Average and Holt-Winters (Triple Exponential Smoothing) for forecasting.	Web-based and component-based applications.	They forecasted performance failures, caused by software aging on the web-based and component-based applications.	Goodness of fit of these models is not discussed in this work.
Yan and Guo 2016	Naive Bayes, Decision Trees, and Artificial Neural Network.	IIS web server.	They developed a method to predict software aging.	The analysis of time to rejuvenation is not done.

self-healing techniques. This technique continuously monitors system resource; if the system resource falls below a certain threshold, then an automatic rejuvenation mechanism is enabled. The rejuvenation mechanism is based on virtualization architecture. The application server is running on a server virtualization platform, and the VM in which application server runs will be migrated to another server in the case of aging effect. Authors tested it with a set of open-source Linux tools and the XEN virtualization middleware. They conducted an experimental study with two application benchmarks (Tomcat/Axis and TPC-W). They monitored several system parameters like CPU, memory usage, swap space, disk usage, the number of threads, I/O traffic, connection to the database, etc.. No specific statistical tools or methods were used in their study.

Silva et al. 2006 developed a method for evaluating and comparing the robustness of some popular SOAP-RPC implementations that used in the industry; the study was on Apache Axis where the aging effect was observed. They developed SLA-oriented software rejuvenation technique that increased the dependability of the SOAP-server, maintained sustainability in the performance of the applications. They compared four java based middleware technologies raw TCP-IP socket, Java RMI, HTTP-XML: using Java Servlets and XML, SOAP-RPC (Tomcat and Axis v1.3). No specific statistical techniques and machine learning algorithms were used to forecast the system resources.

Matias and Filho 2006 studied the aging effects in the Apache web server; further they evaluated the importance of rejuvenation. A software rejuvenation agent was implemented by them to analyze aging effects on the web server. They used Design of experiment (DOE) methods to determine the aging factors and its degree of impact; page size, page type and requests per second rate were monitored. The effectiveness of these methods was not discussed in their study.

Araujo et al. 2011 used a hybrid approach which combines time-series analysis with the threshold-based approach.The study was conducted in the Eucalyptus cloud computing infrastructure; they adopted four models namely Linear Trend Model (LTM), Quadratic Trend Model (QTM), Growth Curve Model (GCM), and S-Curve

Trend Model (SCTM). They monitored memory-related resource consumption of the server. Error measures namely MAPE (Mean Absolute Percentage Error), MAD (Mean Absolute Deviation), and MSD (Mean Squared Deviation) were adopted for choosing the model that best fits monitored data. But volatility and structural breaks in the data were not considered.

Avritzer and Weyuker 2004 developed a simulation model to evaluate performance of E-commerce application. Using this simulation model they evaluated the impact of factors like the application server heap size, the number of application servers that are executing at a certain node, kernel overhead, and the quality of service enforcement algorithm on the performance of E-commerce application. They used weblogs of and E-commerce website for performance testing, and simulation modeling. But there was no comparison made with other measurement-based techniques to evaluate the effectiveness of simulation. Table 2.5 summarizes the merits and demerits of Threshold based methods.

2.2.3 Hybrid Techniques

Hybrid studies combine benefits of both model and measurement-based techniques. Even though it is practically relevant, no considerable attempts were made in this direction, this may be due to the lack of expertise and huge learning curve associated with such models. In this section, we consider some relevant works in this direction.

Vaidyanathan and Trivedi 2005 used UNIX operating system to study software aging and further they demonstrated the relation between the system workload and resource exhaustion. They developed SNMP (Simple Network Management Protocol)-based resource monitoring tool for their data collection. They used system variables like the number of CPU Context Switch, system Call, page-in, and page-out for characterizing the workload and futher they used k-means clustering algorithm to identify a small number of representative workload states.They developed a Semi-Markov process to describe the system workload from the states identified. They considered only memory related resources and they did not model the distribution of time to failure.

Table 2.5: Merits and Demerits of Threshold based Methods

Works	Model Used	Type of System	Merits	Demerits
Silva et al. 2009	They monitored several system parameters like CPU, memory usage, swap space, disk usage, the number of threads, I/O traffic, connection to the database etc.	Application Servers.	Developed a method of software rejuvenation for application servers and it is based on automated self-healing techniques.	No specific statistical tools or methods are used.
Silva et al. 2006	They monitored system parameters. No specific statistical techniques and machine learning algorithms are used.	SOAP-server.	Developed a method for evaluating and comparing the robustness of some popular SOAP-RPC implementations that used in the industry.	No specific statistical techniques and machine learning algorithms to forecast the system resources.
Matias and Filho 2006	Design of experiment (DOE) methods.	Apache web server	Studied the aging effects in the Apache web server. Further, evaluated the importance of rejuvenation.	The effectiveness of these methods is not discussed.
Araujo et al. 2011	Linear Trend Model (LTM), Quadratic Trend Model (QTM), Growth Curve Model (GCM), and S-Curve Trend Model (SCTM).	Eucalyptus cloud computing infrastructure	Developed a hybrid approach which combines time-series analysis with the threshold-based approach.	Volatility and structural breaks in the data are not considered.
Avritzer and Weyuker 2004	Simulation model.	E-commerce application.	Developed a simulation model to evaluate performance of ecommerce applications.	No comparison is made with other measurement-based techniques to evaluate the effectiveness of simulation.

Another significant work using hybrid technique is from Eto et al. 2008, where an adaptive non-parametric estimation scheme was developed for triggering the software rejuvenation. They used reinforcement learning algorithm, called Q-learning for this purpose. Their estimation technique does not require the complete knowledge on system failure (degradation) time distribution. They examined its asymptotic properties through a simulation experiment. No specific system was mentioned in their study.

Hoffmann et al. 2007, integrated the best practices from the experiences of two different studies, namely: the first study was on how the selection of variables contributes more to model quality than selecting a particular modelling technique. In the second study, they compared five linear and non-linear models and found that the superiority of non-linear model is not always significant while comparing the model complexity. They proposed a coherent approach by integrating the goodness of the above studies. They substantiated the effectiveness of the method by modeling and predicting the response time, and the amount of free physical memory of an Apache web server system, as well as the call availability of an industrial telecommunication system. The modeling techniques used are a multivariate linear regression (ML) and support vector machines (SVM). But, volatility and structural breaks in the dataset were not considered in their work.

Yan et al. 2014, proposed a hybrid model that combines autoregressive integrated moving average (ARIMA) and artificial neural network (ANN) to improve the prediction accuracy of resource consumption of an IIS web server which suffered from software aging problems. Their assumption was that the error term was nonlinear in nature and they have used the ANN to model it. But they did not discuss the goodness of the fit for the linear component using ARIMA model and further they did not throw light on the residual analysis. Table 2.6 summarizes the merits and demerits of Hybrid Techniques.

Table 2.6: Merits and Demerits of Hybrid Techniques

Works	Model Used	Type of System	Merits	Demerits
Vaidyanathan and Trivedi 2005	Linear regression, K-means clustering, Semi-Markov process	UNIX Operating System.	Developed a hybrid method for software aging studies of UNIX operating system and demonstrated the relation between the system workload and resource exhaustion.	Considered only memory related resources and no modeling of distribution of time to failure.
Eto et al. 2008	Q-learning, simulation.	No specific system.	Developed an adaptive non-parametric estimation scheme for triggering the software rejuvenation.	No specific system is mentioned in the model studies.
Hoffmann et al. 2007	Multivariate linear regression (ML) and support vector machines (SVM).	Apache web server, Industrial telecommunication system	Integrated best practices from previous experiences. It gives guidelines for future studies.	Volatility and structural breaks in the dataset are ignored.
Yan et al. 2014	Auto-Regressive Integrated Moving Average (ARIMA) and Artificial Neural Network (ANN).	IIS web server	Improved the prediction accuracy of resource consumption of an IIS web server which suffered from software aging problems.	No discussion on the goodness of the fit for the linear component using ARIMA model and did not throw light on the residual analysis.

2.3 Resource Consumption Analysis

Resource consumption analysis is essential in any software performance degradation studies because the primary reason for the performance degradation and aging failures are due to the resource crunch. The reason for the resource crunch is leakage of this resource due to unidentified bugs (Mandelbugs). The software aging studies can be classified based on the system resources analyzed. Here we will classify the studies into two: memory related resources and other system related resources. Table 2.7 summarizes the merits and demerits of resource consumption based techniques.

2.3.1 Memory Related Resources

Memory related resources usage analysis is the most popular analysis among the software performance degradation studies, as it exhibits the shortest time to exhaustion. Garg et al. 1998b analyzed the swap space and free memory. They applied seasonal Kendall test to each of these time series to detect the presence of aging patterns and applied linear regression methods to estimate the true slope of the trend. Li et al. 2002, Grottke et al. 2006 developed a method based on Auto-Regressive Moving Average (ARMA) model for estimating resource exhaustion in a Web server. They collected response time, free physical memory and used swap space of an Apache web server and applied Man-Kendall test to determine performance degradation.

Cotroneo et al. 2010 studied software aging in a Linux Kernel by monitoring memory related like page misses, page cache flushes, cache insertions, interrupt requests, etc. They used Mann-Kendall test for trend detection. Multiple Linear Regression models are used to study the relation between the aging and system variables and used principal component analysis (PCA) to avoid multicollinearity.

Cotroneo et al. 2013 analyzed software aging in a Java Virtual Machine (JVM). They found some aging effects like throughput loss and memory leakage in both Just-in-Time (JIT) compiler and garbage collector. They used hypothesis testing to establish aging trend and used multiple linear regression to estimate the aging trend. The major drawback in the memory related resource consumption studies

was that the memory related resources exhibited volatility and structural especially when they start aging, but researchers have not given much attention to this fact.

2.3.2 Other System Resources

Zhang et al. 2011 developed a method called MODE, "Mine Once, Detect Everywhere" to detect potential resource leaks automatically before code check-in. They evaluated the effectiveness of their method; they developed Java API call tracer based on Java Virtual Machine Tool Interface (JVMTI) and applied this on three open source projects in Apache.

Weimer 2006 analyzed the behavior of computer programs in the presence of exceptions and developed a method for finding a certain class of mistakes while programs handle exceptions. This method depends on the data flow analysis, using this mechanism they found over 1,200 procedures with mistakes in almost 4 million lines of Java code. The existing language features were insufficient to handle such mistakes, so they designed a language feature to make it easier to fix such mistakes. Garg et al. 1998b used file table size to analyse aging trend in web server, and more details are given in section 2.2.2.

2.4 Outcome of Literature Review

In the literature review, we first reviewed different dimensions in the research area of software performance/software aging and rejuvenation studies, namely: type of analysis, type of systems and resource consumption analysis. The type of analysis is broadly classified into three: model-based, measurement-based, and hybrid techniques. The model-based techniques can be applied to the system at any stage of the product lifecycle, but the measurement based methods can be applied only in the post-prototype stage of the system under study because it requires a real system for measuring the aging indicators/resources. The time required for applying a model-based technique is comparatively lower than other two methods, namely: measurement based techniques and hybrid techniques. The cost of model-

Table 2.7: Merits and Demerits of Methods based on Resource Consumption Analysis

Works	Resources Analyzed	Type of System	Merits	Demerits
Garg et al. 1998b	Swap Space, Free Memory, /tmp Directory usage and File Table Size	UNIX operation system.	Analyzed the memory related resources by taking into account the file related resource usage.	The data shows structural breaks, but this is not taken into account during data analysis.
Li et al. 2002,Grottke et al. 2006	Memory, file and thread related resources; Response Time	Apache web server.	Designed a method for estimating memory related resource exhaustion in a Web server.	The data shows heteroskedasticity, but this is not taken into account during data analysis.
Cotroneo et al. 2010	Memory-related, file-related, process-related, and device driver-related resources	Linux Operating System.	Focused more on Memory related and file related resources for studying aging effect.	The main focus is on aging effect rather than resource forecasting, so less importance is given to the forecasting models.
Cotroneo et al. 2013	Memory-related Resources	Java Virtual Machine (JVM).	Found some aging effects like throughput loss and Memory leakage in both Just-in-Time (JIT) compiler and Garbage collector.	Goodness of fit of these models is not discussed in this work.
Magalhaes and Silva 2010	Memory, file and thread related resources.	Web-based and component-based applications.	Determined the correlation between the application response time and workload.	Goodness of fit of these models is not discussed in this work.
Zhang et al. 2011,Weimer 2006	Program files.	No specific system.	Detected potential resource leaks automatically before code check-in.Weimer 2006 Developed a proactive method for finding a certain class of mistakes while programs handle exceptions.	Resource forecasting is not done in this work.

based studies is lower as compared to both measurement-based method and hybrid method. The trade-off analysis is also comparatively easier on model-based techniques when compared to the other two techniques. Even in the midst of all advantages, the model-based techniques are lesser-known compared to measurement-based techniques due to the two key reasons, namely: 1) the accuracy of model-based techniques are inferior to that of measurement-based techniques, and 2) the scalability of the results is comparatively lower than that of measurement techniques. The accuracy and the scalability of the results are more or less similar in the case of hybrid and measurement-based techniques, but the difficulty of implementation is high for hybrid technique. The measurement-based techniques are superior to both hybrid and model-based techniques when we consider the cost-benefit analysis. We used measurement-based techniques throughout this thesis work. Table 2.8 summarizes the comparison of three techniques used commonly by the researchers across the globe.

The other dimension we considered during the literature review is the type of the system; we found that virtualized server consolidation systems are less studied when compared to other non-safety critical systems like a web server and operating system. The reason for selecting virtualized server consolidation systems is due to the increased popularity of cloud computing. Resource consumption analysis is essential in virtualized server consolidation system because the resources are used based on demand hence the resource forecasting is challenging. Considering the importance of virtualized server consolidation and because it is a less explored system hence it is decided to explore it further. It is clear from the related work that the most popular models among the researchers studying the resource exhaustion are time series models. Some authors found that these data shows non-linear dynamic patterns like volatility and the structural changes in the data, but most of the authors did not consider the nonlinearity in the data; this motivated us to explore the analysis of volatility and the structural changes in the data. We decided to investigate the suitability of different time series models for the resource consumption analysis of an aged virtualized server consolidation system in this thesis.

The last dimension is the type of resource used for consumption analysis. From the literature review, we observed that memory related resources usage analysis is the most popular among the software performance degradation studies, as it exhibits the shortest time to exhaustion. We used memory related resource for our consumption analysis throughout this thesis.

2.5 Problem Statement

The aim of this research thesis is to analyze the resource consumption, mainly memory consumption of a virtualized server consolidation system. The problem statement is as follows:

"To investigate the suitability of different time series models for the resource consumption analysis of an aged virtualized server consolidation system".

2.6 Research Objectives

- To establish aging effect in the resource usage data collected from server consolidation system. To examine the effect of aging on the power usage and percentage CPU utilization when the virtualized server consolidation system is in a probable failure state.
- To analyze the forecasting errors associated with ARIMA models and examines the presence of heteroscedasticity.
- To understand the prediction errors associated with non-linear and heteroscedastic models and carry out the residual analysis for any further improvements.
- To investigate the presence of structural breaks and also to scrutinize the appropriateness of regime switching models for resource consumption analysis.

Table 2.8: Comparison of Model-based, Measurement-based and Hybrid Techniques

Criterion	Model-based Techniques	Measurement-based Techniques	Hybrid Techniques
Stage of the product Life Cycle	Applied at any stage.	Post Prototype.	Post Prototype.
Time Required	Small.	Experimentation time is large.	Experimentation time is large.
Tools Required	Analysts.	Instrumentation.	Depends on modeling techniques.
Accuracy	Low.	High.	High.
Cost	Low.	High.	Varies.
Scalability	Low.	High.	Varies.
Implementation	Easy.	Medium.	Difficult.

2.7 Summary

This chapter reviewed all the major state-of-the art works in the area of software aging and rejuvenation. The broad categorization of these works are illustrated in Figure 2.1. The observations from the relevant category are summarized in the Tables 2.1 to 2.8. The chapter concluded with an outcome of the literature review followed by problem statement and objectives. The next chapter will consider the first two research objectives, i.e., to establish aging effect in the resource usage data collected from server consolidation system and to analyze the forecasting errors associated with ARIMA models and examines the presence of heteroscedasticity.

Chapter 3

Aging Trend Detection and Resource Forecasting with ARIMA Model

The primary focus of this chapter is to establish aging effect in the resource usage data collected from virtualized server consolidation system. The final focus of this chapter is to analyze the forecasting errors associated with ARIMA models, thereby examining the presence of heteroscedasticity. The details of aging trend detection are as follows.

3.1 Aging Trend Detection

To establish the aging effect, we considered the following steps,

- Collect the resource usage data for a longer time period, the length of this time period depends on the computational workload, the physical capacity of the server, and the nature of the software system.
- The data collected has to be modeled with a simple linear regression model.
- The model which is developed will be used for hypothesis testing for analyzing the aging effect.

The overall method can be depicted in Figure 3.1.

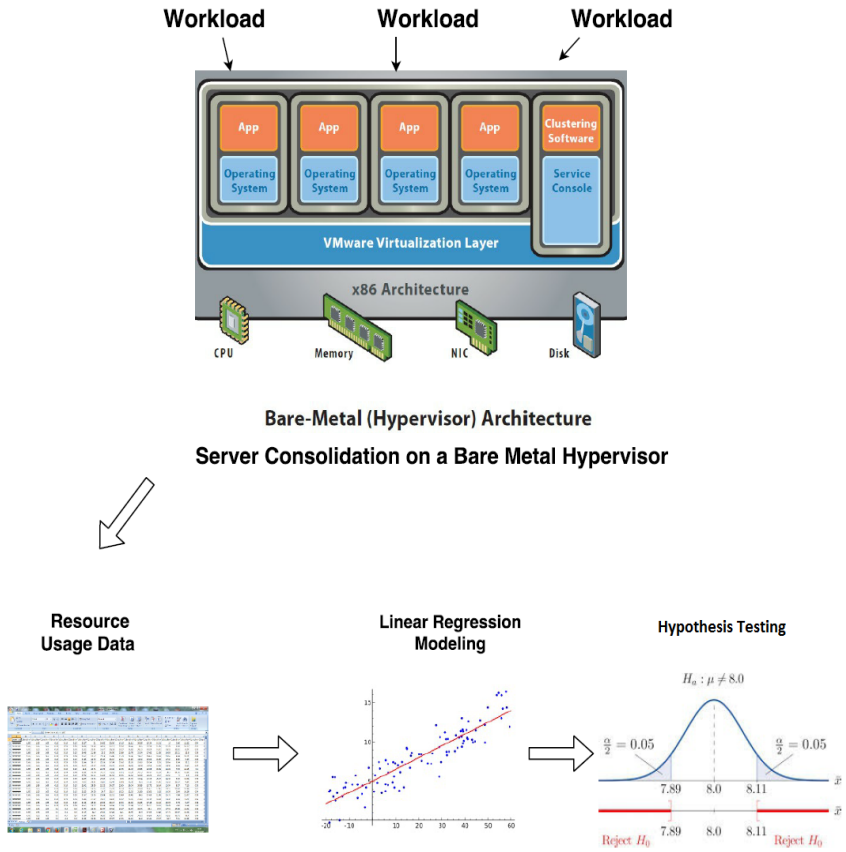

Figure 3.1: Method to Identify Aging Effect

3.1.1 Experimental Setup

We conducted experiments on HP ProLiant ML350 G6 machine; this server has two Intel Xeon architecture processors with six cores in each processor with CPU clock speed of 2.40 GHz. When hyperthreading enabled, this server effectively has 24 logical processors. This server has one terabyte hard disk space and sixteen gigabyte main memory (RAM) and it has 2 network interface controller cards to manage the network traffic effectively. We used Vmware ESXi 5.0.0 as hypervisor in this experiment; this hypervisor/VMM belongs to TYPE 1 VMM. We installed Apache Tomcat 6.0.36 web server on the guest operating system. The experimental setup is as shown in Figure 3.2. We used *httpref* tool for generating the requests with constant time intervals between two requests, that is the workload, each *http*

request accesses one specified file of size five kilo bytes from the web server. In this experiment, *httpref* is considered as a workload generator, and this tool also gives performance metrics. These metrics involve reply rate, response time and timeout rate; we monitored only the response time. We used *esxtop* command to collect the measures like CPU usage, Disk I/O usage, Swap space usage, the number of interrupts calls, the number of context switches, bandwidth usage, and power usage. For the said purpose, we edited the configuration file of the esxtop command.

There are four memory reclamation techniques (VMware) available in VMware ESXi 5.0.0, namely: 1) Transparent Page Sharing (TPS), 2) Ballooning, 3)Hypervisor swapping, and 4) Memory compression. Transparent page sharing removes redundant pages with identical content and consolidates to a single page while ballooning recovers memory by artificially increasing the memory pressure inside the guest. In memory compression, ESXi stores the pages by compressing, otherwise these pages would have been swapped out to disk through host swapping. These compressed pages are stored in a compression cache located in the main memory. Memory compression outperforms host swapping because the next access to the compressed page only causes a page decompression. The page decompression is faster than the disk access. In the cases where ballooning, transparent page sharing, and memory compression are not sufficient to reclaim memory, ESXi employs hypervisor swapping to reclaim memory. Hypervisor swapping guarantees the reclamation of a specific amount of memory within a specific amount of time. However, hypervisor swapping is associated with performance limitation, hence it is used as the last way to reclaim memory from the virtual machine.

3.1.2 System Resources Monitored

In this work, we monitored mainly memory related resources; as mentioned earlier it exhibits the short time to exhaustion. This section briefly goes through the resources we monitored and its significance.

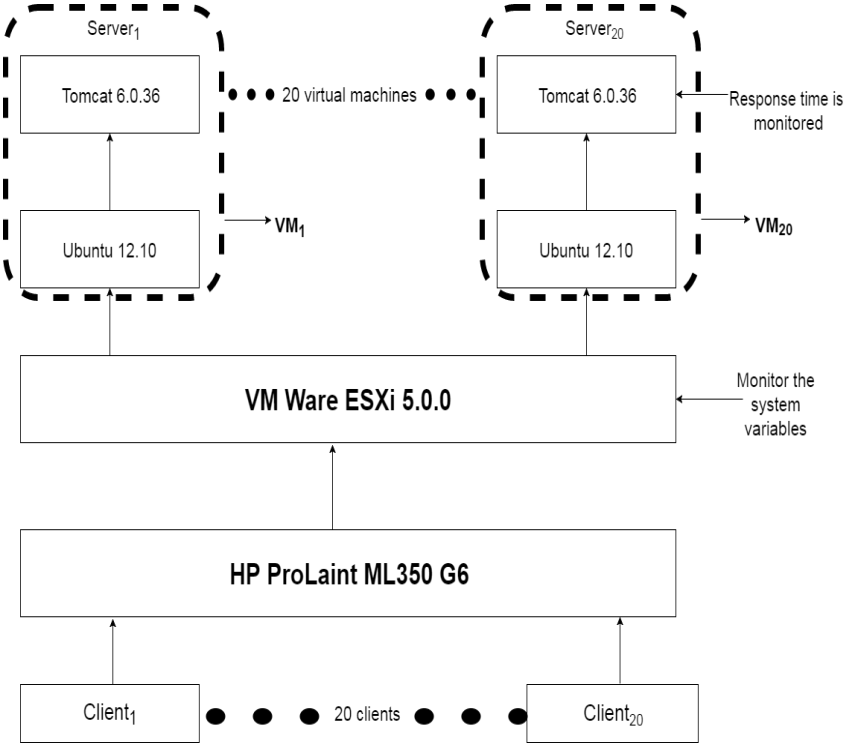

Figure 3.2: Experimental Setup

Memory Resources

- **Free MBytes:** This is the total memory available for VMM.
- **Memory Overcommit (1, 5, 15 Minute Avg):** Memory overcommit is the ratio of total requested memory and the "managed memory" minus 1. VMKernel computes the total requested memory as a sum of the following components: (a) VM configured memory (or memory limit setting if set), (b) the user world memory, (c) the reserved overhead memory. The "managed memory" is the total amount of machine memory managed by VMKernel. VMKernel "managed" memory can be dynamically allocated for VM, VMKernel, and User Worlds.
- **PShare Shared MBytes:** This is the amount of guest physical memory that is being shared.
- **PShare Savings MBytes:** This is the amount of machine memory saved due to page sharing.

- **PShare Common MBytes:** The amount of machine memory that is common across the world. The world is an ESX Server VMkernel schedulable entity, similar to a process or thread in other operating systems. A group contains multiple worlds.
- **Machine MBytes:** The total MBytes available in the machine.
- **Swap Used MBytes:** Used Swap Space in MBytes.
- **Swap MBytes Read/sec:** Swap Space read per sec in MBytes.
- **Swap MBytes Write/sec:** Swap Space write per sec in MBytes.
- **Total Compressed MBytes:** Total memory compressed by the memory compression reclamation technique.
- **Total Saved By Compression MBytes:** Total memory saved by the memory compression reclamation technique.
- **Kernel Reserved MBytes:** This is the memory reserved for the kernel.
- **Kernel Unreserved MBytes:** This is the memory available for effective usage.

CPU Resources

- **% Core Util Time:** This is displayed only when hyper-threading is enabled. This is the average % of CPU cycles per core when at least one of the Physical Central Processing Units (PCPUs) in this core is not halted.
- **CPU Load (1,5,15 Minute Avg):** The arithmetic mean of CPU loads in 1 minute, 5 minutes, and 15 minutes.

Power Usage

- **Power Usage Now Watts:** This is the power usage in watts by the machine at the time of monitoring.

3.1.3 Linear Regression Modeling

A linear regression model is a mathematical representation of the relationship between the response variable and the regressor variable. Linear regression models are widely used to obtain estimates of parameter significance as well as predictions of the response variable at arbitrary points in the design space. One of the simpler forms of such model is given by Eq.(3.1).

$$y = \beta_0 + \beta_1 x + \epsilon \tag{3.1}$$

Where y is the dependent or response variable and x is the independent or regressor variable. The parameters β_0 and β_1 are usually called the regression coefficients. ϵ is the error associated with the prediction.

The parameters β_0 and β_1 are estimated using method of least square estimation. The least square estimators of β_0 and β_1, are $\hat{\beta}_0$ and $\hat{\beta}_1$ respectively. The least square estimators are obtained by the following equations (Eq.(3.2), Eq.(3.3)).

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{3.2}$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - \frac{(\sum_{i=1}^{n} y_i)(\sum_{i=1}^{n} x_i)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{(\sum_{i=1}^{n} x_i)^2}{n}} \tag{3.3}$$

Where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ and $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ are the averages of y and x, respectively. The fitted simple linear regression model is then given by Eq. (3.4).

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_i \tag{3.4}$$

As per Gauss-Markov theorem, the regression model thus obtained with the least square estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ follows certain assumptions about the error random variables,ϵ_i.

- $E(\epsilon_i) = 0$ **:** The errros have a mean of zero.
- $Var(\epsilon_i) = \sigma^2 < \infty$ **:** The errors are homoscedastic with the same finite variance.
- $Cov(\epsilon_i, \epsilon_j) = 0; \forall i \neq j$ **:** Distinct error terms are uncorrelated.

Coefficient of Determination

The Coefficient of Determination or R^2 pronounced as "R-squared" is the proportion of the variation explained by the regressor x. R^2 will give some information about the goodness of fit of a model. In a simple linear regression, R^2 tells how well the regression line approximates the real data points. R^2 ranges between zero and one; if R^2 is one then it indicates that the regression line perfectly fits the data. The Coefficient of Determination is given by Eq.(3.5).

$$R^2 = \frac{SS_R}{SS_T} \tag{3.5}$$

where $SS_T = SS_R + SS_{Res}$. SS_T is the total sum of squares , SS_R is the regression sum of squares, and SS_Res is the sum of squares of residuals. And SS_T, SS_R, and SS_{Res} are given by Eqs. (3.6, 3.7, and 3.8) respectively.

$$SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{3.6}$$

$$SS_R = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \tag{3.7}$$

$$SS_{Res} = \sum_{i=1}^{n} (y_i - \hat{y})^2 \tag{3.8}$$

Where y_i, $\bar{y}$, and $\hat{y}_i$ are observed, average and predicted values of the response variable y, respectively.

3.1.4 Testing Significance of Regression

To test the significance of the regression $y = \beta_0 + \beta_1 x_1 + \epsilon$ the following hypothesis (Eq. 3.9) has to be considered.

$$\begin{aligned} H_0 &: \beta_1 = 0 \\ H_1 &: \beta_1 \neq 0 \end{aligned} \tag{3.9}$$

Failing to reject $H_0 : \beta_1 = 0$ implies that there is no significant linear relationship between the response variable y and the regressor variable x. We used three techniques to test the significance of regression. The first approach is to use $t-Test$, second is the analysis of variance and third is the confidence interval of β_1.

Using t-Test

To test the hypothesis $H_0 : \beta_1 = 0$ we used the t-statistic given by Eq. (3.10).

$$t_0 = \frac{\hat{\beta}_1}{se(\hat{\beta}_1)} \tag{3.10}$$

Where the standard error $se(\hat{\beta}_1)$ is given by Eq. (3.11).

$$se(\hat{\beta}_1) = \sqrt{\frac{MS_{Res}}{S_{xx}}} \tag{3.11}$$

The unbiased estimator of population variance σ^2 also called as the residual mean square MS_{Res} is given by Eq. (3.12). The S_{xx} is given by Eq. (3.13).

$$MS_{Res} = \frac{SS_{Res}}{n-2} \tag{3.12}$$

$$SS_{xx} = \sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{3.13}$$

The null hypothesis $H_0 : \beta_1 = 0$ of the regression model $y = \beta_0 + \beta_1 x_1 + \epsilon$ is rejected if $|t_0| > t_{\frac{\alpha}{2},n-2}$.

Using Analysis of Variance

We further used analysis of variance (ANOVA) to test the significance of regression model $y = \beta_0 + \beta_1 x_1 + \epsilon$. To test hypothesis $H_0 : \beta_1 = 0$, we used the F statistic given by Eq. (3.14).

$$\begin{aligned} F_0 &= \frac{\frac{SS_R}{df_R}}{\frac{SS_{RES}}{df_{RES}}} \\ &= \frac{\frac{SS_R}{1}}{\frac{SS_{RES}}{n-2}} \end{aligned} \tag{3.14}$$

The model or regression sum of squares, SS_R, has $df_R = 1$. The degree of freedom of sum of sqaures of residuals, SS_{RES}, is $df_{RES} = n - 2$. To test the hypothesis $H_0 : \beta_1 = 0$, we have to compute F_0 and reject the H_0 if $F_0 > F_{\alpha,1,n-2}$. Where α gives the confidence level of the test.

Confidence Intervals of β_1

The significance of the null hypothesis $H_0 : \beta_1 = 0$ of the regression model $y = \beta_0 + \beta_1 x_1 + \epsilon$ can be checked by looking at the confidence interval of β_1. If the errors of the regression are normally and independently distributed, then the sampling distribution of $\frac{\hat{\beta}_1 - \beta_1}{se(\hat{\beta}_1)}$ follows a t distribution with $n - 2$ degrees of freedom. Hence, a $100(1 - \alpha)$ percent confidence interval on the slope β_1 is given by Eq. (3.15).

$$\hat{\beta}_1 - t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \leq \beta_1 \leq \hat{\beta}_1 + t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \tag{3.15}$$

If '0' do not fall in this range of β_1 given by Eq. (3.15), we may reject the null

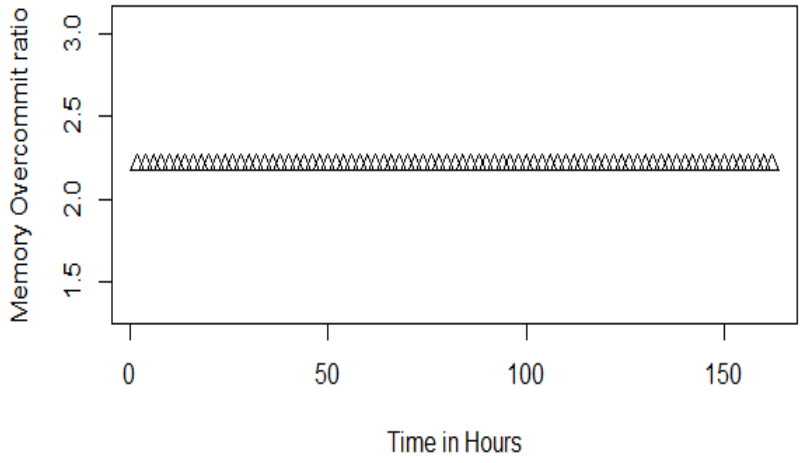

Figure 3.3: Memory Overcommit (15 Minute Average)

hypothesis $H_0 : \beta_1 = 0$ of the regression model $y = \beta_0 + \beta_1 x_1 + \epsilon$.

3.1.5 Result Analysis of Aging Trend Detection

As per section 3.1.1, we have setup an experiment and collected the data for 162 hours. The system resources monitored were discussed in section 3.1.2. As this is an experiment to detect the aging trend detection, we ensured that we have collected data till the system shows the performance degradation. The memory reclamation techniques like transparent page sharing and memory compression techniques were enabled, and other techniques like ballooning and hypervisor swapping are disabled. The reason for disabling the last two techniques is to make a balance between this rejuvenation mechanism if all the techniques were enabled the time to show the aging effect will be longer than the currently observed one. Another reason to disable balooning is that an extra driver has be installed in the VM to enable the balooning it may be an additional computaional overhead. The memory overcommit for 5, 10, and 15 minutes average have been kept over 2.1 to ensure enogh memory intensive workload as shown in Figure. 3.3

The memory saved due to Transparent Page Sharing (TPS) and Memory Com-

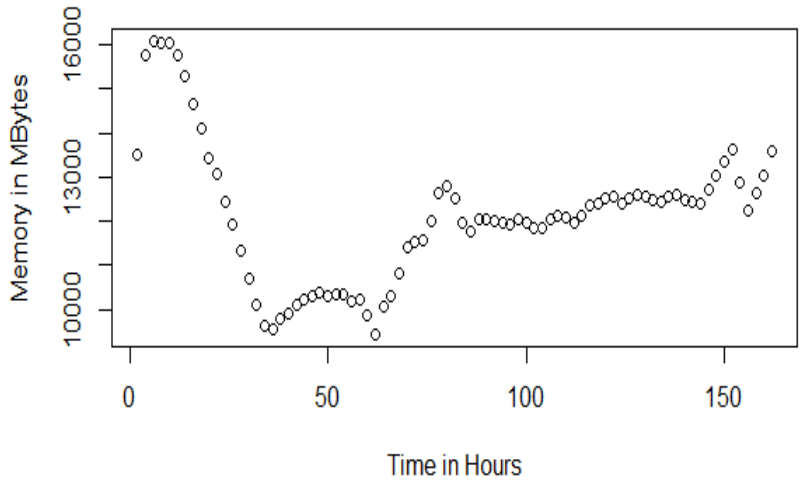

Figure 3.4: Memory saved by Transparent Page Sharing (TPS) in Mbytes Against Time in Hours

pression is shown in Figures 3.4 and 3.5 respectively. Figures. 3.4 and 3.5 demonstrate that, as the time progresses, memory saved due to pages share and compression becomes stagnant thus keeping the virtualized server consolidation system healthier. This memory saving process depends on the nature of workload and a prediction on the size of memory saved in Mbytes by memory reclamation techniques is not feasible.

In order to know any performance degradation in the web servers, loaded on top of the geust operating system as shown in the experimental setup (Figure 3.2),we plotted a graph (Figure 3.6) which gives the average response time of 20 web severs in milliseconds against time in hours. Eq. (3.16) gives the least square fitted linear regression model for the average response time of the 20 webservers as the dependent variable and elapsed time as the independent variable.

$$\hat{y}_i = 107.38 + 2.11x_i \tag{3.16}$$

From the coefficient of time, it is found that there is 2.11 milliseconds increase per hour in the average response time of web server. In order to establish the significance

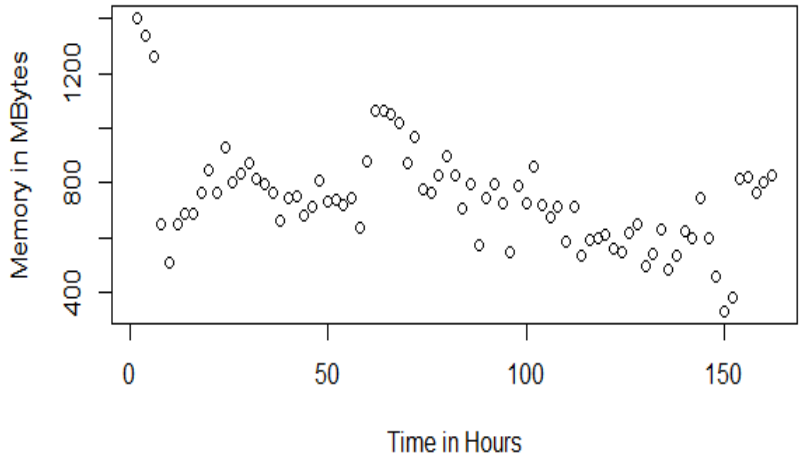

Figure 3.5: Memory saved by Memory Compression Technique in Mbytes Against Time in Hours

of linear regression (Eq. (3.16)), we considered the null hypothesis $H_0 : \beta_1 = 0$ which conveys that there is no relevant correlation between time and average response time. We determined t-static from the sample data, which is $|t_0| = 126.849$ at $\alpha = 0.05$, the value of t-static is $t_{0.025,160} = 1.645$. Hence the null hypothesis $H_0 : \beta_1 = 0$ of the regression model $\hat{y}_i = 107.38 + 2.11x_i$ is rejected as $|t_0| > t_{\frac{\alpha}{2},n-2}$.

We used analysis of variance approach to affirm significance of regression. To check the null hypothesis $H_0 : \beta_1 = 0$, we determined F_0 from the sample data. Here $F_0 = 16090.58$ for $\alpha = 0.05$; $F_{0.05,1,160} = 2.75$ which shows that $|F_0| > F_{\alpha,1,n-2}$. So we rejected the null hypothesis.

A 95 percentage confidence interval of the coefficient of time, that is β_1 is shown in Eq. (3.17). This shows that the '0' do not fall in this range of β_1 given by Eq. (3.17), we may reject the null hypothesis $H_0 : \beta_1 = 0$ of the regression model $\hat{y}_i = 107.38 + 2.11x_i$.

$$\begin{aligned} \hat{\beta}_1 - t_{\frac{\alpha}{2},n-2}se(\hat{\beta}_1) \leq \beta_1 \leq \hat{\beta}_1 + t_{\frac{\alpha}{2},n-2}se(\hat{\beta}_1) \\ 2.078706992 \leq \beta_1 \leq 2.144983162 \end{aligned} \tag{3.17}$$

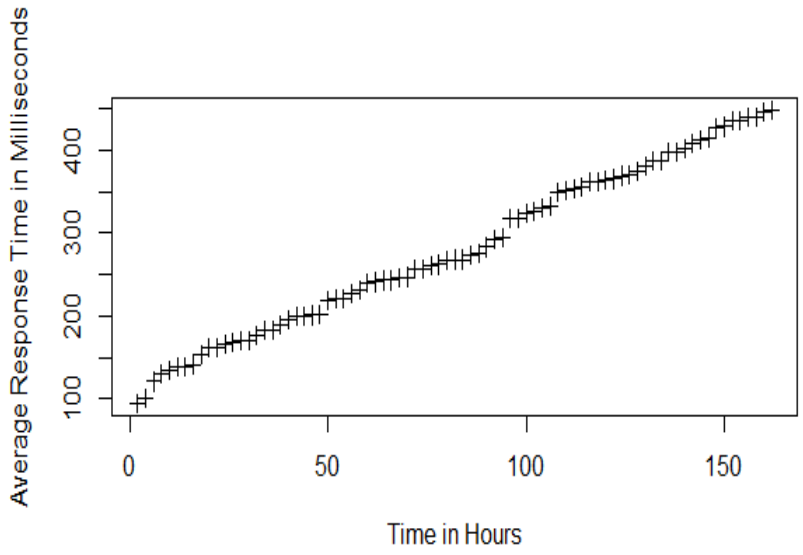

Figure 3.6: The Average Response Time of 20 Web Servers in Milliseconds Against Time in Hours

The Coefficient of Determination or R^2 is 0.90, so 90% of the variability of response time is explained by the regressor x, i.e. time elapsed.

Similarly, the free memory or available memory in Mbytes, % CPU utilization time, and power used in watts are plotted against time as shown in Figures 3.7, 3.8, and 3.9 respectively.

The Eqs. (3.18), (3.19), (3.20) give the least square fitted simple linear regression models for free memory in Mbytes, % CPU Utilization, and power usage in watts as the dependent or response variable y, and time elapsed as the independent or regressor variable x, respectively.

$$\hat{y}_i = 2756.52 - 7.83x_i \tag{3.18}$$

$$\hat{y}_i = 47.05 + 0.38x_i \tag{3.19}$$

$$\hat{y}_i = 161.28 + 0.35x_i \tag{3.20}$$

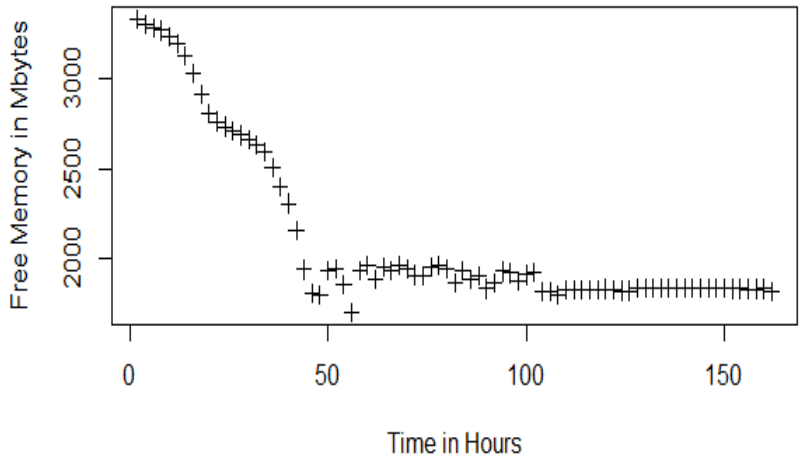

Figure 3.7: The Available Memory in Mbytes Plotted Against Time in Hours

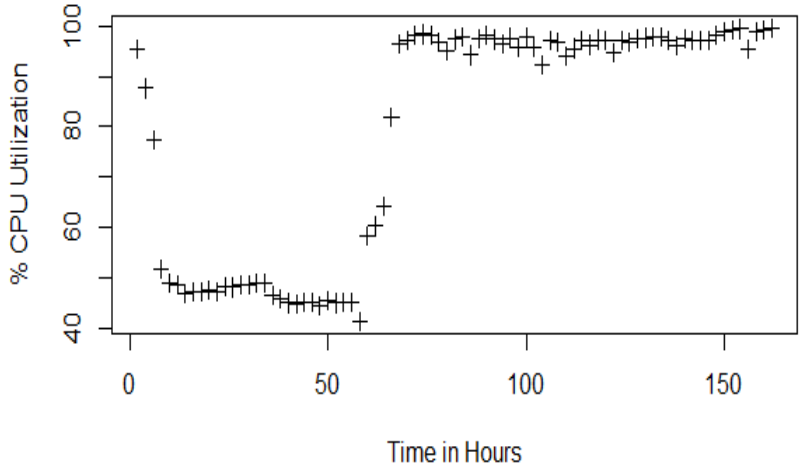

Figure 3.8: % CPU ultilization Plotted Against Time in Hours

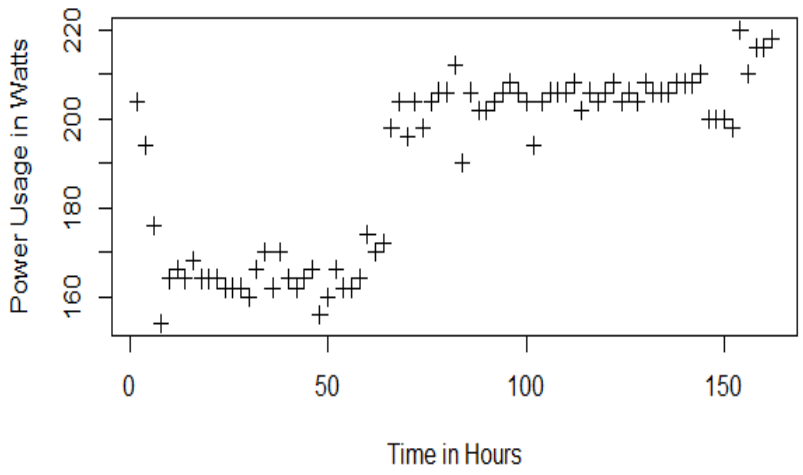

Figure 3.9: Power Usage in Watts Plotted Against Time in Hours

From β_1 of Eq. (3.18), it is found that there is 7.83 Mbytes decrease in free memory per hour. In order to establish the significance of regression, we considered the null hypothesis $H_0 : \beta_1 = 0$ which conveys that there is no relevant correlation between the time in hours and free memory, we determined t-static from the sample data. $|t_0| = 11.14$ at $\alpha = 0.05$, the value of t static is $t_{0.025,160} = 1.645$. The null hypothesis $H_0 : \beta_1 = 0$ of the regression model $\hat{y}_i = 2756.52 - 7.83x_i$ is rejected as $|t_0| > t_{\frac{\alpha}{2},n-2}$. We considered the ANOVA to reaffirm the significance of regression. To test the significance null hypothesis $H_0 : \beta_1 = 0$ of Eq. (3.18), we calculated F_0 from the sample data. Here $F_0 = 124.22$ for $\alpha = 0.05$; $F_{0.05,1,160} = 2.75$ which shows that $|F_0| > F_{\alpha,1,n-2}$. Since $|F_0| > F_{\alpha,1,n-2}$ we rejected the null hypothesis. A 95 percentage confidence interval of the coefficient of time of Eq. (3.18), that is β_1 is shown in Eq. (3.21). This shows that the '0' do not fall in this range of β_1 given by Eq. (3.21), we may reject the null hypothesis $H_0 : \beta_1 = 0$ of the regression model $\hat{y}_i = 2756.52 - 7.83x_i$

$$\begin{aligned} \hat{\beta}_1 - t_{\frac{\alpha}{2},n-2}se(\hat{\beta}_1) \leq \beta_1 \leq \hat{\beta}_1 + t_{\frac{\alpha}{2},n-2}se(\hat{\beta}_1) \\ -9.23 \leq \beta_1 \leq -6.43 \end{aligned} \tag{3.21}$$

The Coefficient of Determination or R^2 is 0.61 for the linear model of Eq. (3.18), hence 61% of the variability of free memory is explained by the regressor x, i.e. time elapsed.

From β_1 of Eq. (3.19), it is found that there is 0.38% increase in % CPU utilization per hour. To establish the significance of regression we considered the null hypothesis $H_0 : \beta_1 = 0$ which conveys that there is no significant correlation between time in hours and % CPU utilization, we calculated t-static from the sample data. $|t_0| = 10.98$ at $\alpha = 0.05$, the value of t-static is $t_{0.025,160} = 1.645$. The null hypothesis $H_0 : \beta_1 = 0$ of the regression model given by Eq. (3.19) is rejected as $|t_0| > t_{\frac{\alpha}{2},n-2}$. We considered ANOVA to affirm the significance of regression of Eq. (3.19). To test the null hypothesis $H_0 : \beta_1 = 0$ of Eq. (3.19), F_0 is determined from the sample data. Here $F_0 = 120.58$ for $\alpha = 0.05$; $F_{0.05,1,160} = 2.75$ which shows that $|F_0| > F_{\alpha,1,n-2}$. We rejected the null hypothesis. A 95 percentage confidence interval of the coefficient of time of Eq. (3.19), that is β_1 is shown in Eq. (3.22). This shows that the '0' do not fall in this range of β_1 given by Eq. (3.22), we may reject the null hypothesis $H_0 : \beta_1 = 0$ of the regression model given by Eq. (3.19).

$$\begin{aligned} \hat{\beta}_1 - t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \leq \beta_1 \leq \hat{\beta}_1 + t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \\ 0.31 \leq \beta_1 \leq 0.46 \end{aligned} \tag{3.22}$$

The Coefficient of Determination or R^2 is 0.60 for the linear model of Eq. (3.19), hence 60% of the variability of % CPU utilization is explained by the regressor x, i.e. time elapsed.

Similarly for β_1 of Eq. (3.20), it is found that there is 0.35 watts of increase in power usage per hour. To test the significance of the null hypothesis $H_0 : \beta_1 = 0$ of Eq. (3.20) which states that there is no significant correlation between time in hours and power usage (note: the workload is constant for this period), t-static is determined from the sample data. $|t_0| = 12.40$ at $\alpha = 0.05$, the value of t static is $t_{0.025,160} = 1.645$. The null hypothesis $H_0 : \beta_1 = 0$ of the regression model given by Eq. (3.20) is rejected as $|t_0| > t_{\frac{\alpha}{2},n-2}$. The analysis of variance approach

is considered to reaffirm the significance of regression of Eq. (3.20). To test the null hypothesis $H_0 : \beta_1 = 0$ of the regression Eq. (3.20), F_0 is calculated from the sample data. Here $F_0 = 153.76$ for $\alpha = 0.05$; $F_{0.05,1,160} = 2.75$ which shows that $|F_0| > F_{\alpha,1,n-2}$. Since $|F_0| > F_{\alpha,1,n-2}$, we rejected the null hypothesis. A 95 percentage confidence interval of the coefficient of time of Eq. (3.20), that is β_1 is shown in Eq. (3.23). This shows that the '0' do not fall in this range of β_1 given by Eq. (3.23), we may reject the null hypothesis $H_0 : \beta_1 = 0$ of the regression model given by Eq. (3.20).

$$\begin{aligned} \hat{\beta}_1 - t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \leq \beta_1 \leq \hat{\beta}_1 + t_{\frac{\alpha}{2},n-2} se(\hat{\beta}_1) \\ 0.29 \leq \beta_1 \leq 0.40 \end{aligned} \tag{3.23}$$

The Coefficient of Determination or R^2 is 0.66 for the linear model of Eq. (3.20), hence 66% of the variability of power usage is explained by the regressor x, i.e. time elapsed.

From the above discussion, it is clear that the average response time of the web servers significantly increased over a time period when the workload remained constant for the said time. On further investigation, we found that there is free memory shrinkage during this time. We also found that there is an increase in % CPU utilization and power usage during this period. From the above discussions and results, we can conclude that the performance degradation of web servers, shrinkage of free memory accompanied by the increase in % CPU utilization and power usage is due to the aging-related error propagation or simply due to aging effect.

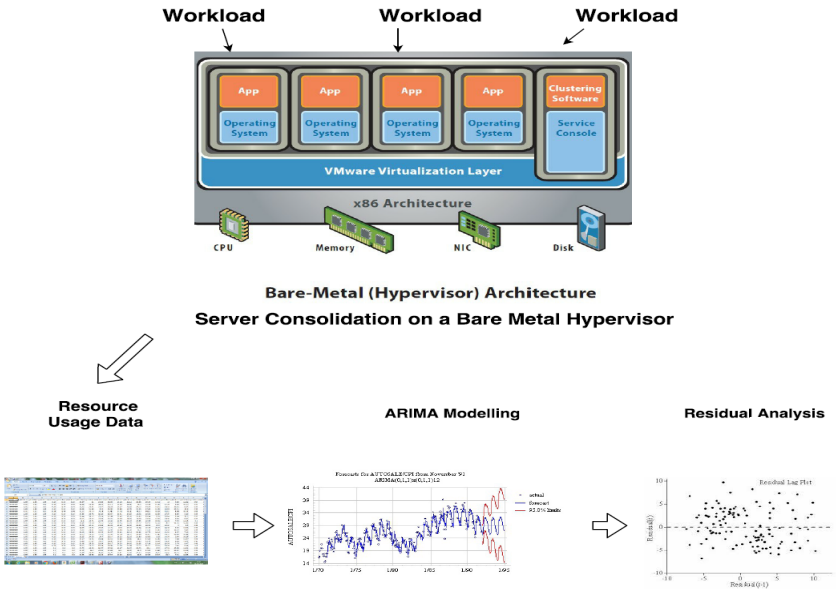

Figure 3.10: Overview of ARIMA Modeling

3.2 Resource Forecasting with ARIMA Model

In this section, we will be discussing the resource usage forecast using ARIMA model. To forecast resource usage with ARIMA, we considered the following steps,

- Collect the resource usage data till the system fails or reaches a near-failure state. The length of this period depends on the computational workload, the physical capacity of the server, and the nature of the software system.
- The data collected has to be modeled with an ARIMA.
- Carry out the residual analysis to check the effectiveness of the model.

Figure. 3.10 illustrates the overall method of resource forecasting using ARIMA.

3.2.1 Experimental Setup

The experimental setup is similar to that of section 3.1.1, here 24 Virtual Machines (VMs) were created on this server and installed with Ubuntu 14.04 operating system in all VMs. Apache 2.0 and PHP 5.0 were installed on these VMs. VMWare ESXi 5.5 was used as the hypervisor. Shell program is written to call the *test.php* page

using *httperf* continuously with the rate of 400 requests per second. The VMs used 1vCPU, 3GB RAM and 50GB hard disk with thin provisioning. 24VMs with 3GB RAM results in 72GB of logical RAM, but physically only 16 GB RAM is present, hence the main memory is over-committed to approximately 4.5 times. A memory leak is deliberately injected to accelerate the failure rate. System resources monitored were discussed in section 3.1.2. All memory reclamation techniques were enabled in this experiment except ballooning.

3.2.2 ARIMA Modeling

An auto-regressive integrated moving average (ARIMA) model is a generalization of an auto-regressive moving average (ARMA) model. Both ARMA and ARIMA models are used to fit the time series data for two reasons, 1) to understand the data better and 2) to forecast the future points in the time series. ARIMA model is applied when data shows traces of non-stationarity, in such cases, we will difference the time series one or more times to eliminate the non-stationarity.

The AR part of $ARIMA$ stands for "auto-regressive" which implies that the variable of interest is regressed on its own prior or lagged values. The MA part stands for "moving average" indicates that the regression error is actually a linear combination of current and previous error terms. The I (for "integrated") indicates the order of differencing needed to make the time-series stationary.

An $ARIMA(p, q, d)$ (Autoregressive Integrated Moving Average with orders p, d, q) can be represented by the Eq. (3.24).

$$\left(1 - \sum_{k=1}^{p} \alpha_k B^k\right)(1 - B)^d X_t = \left(1 + \sum_{k=1}^{q} \beta_k B^k\right)\epsilon_t \tag{3.24}$$

Where X_t is the time series, α and β are the parameters/coefficients of autoregressive and moving average terms with order p and q respectively. ϵ_t are error terms generally assumed to be independent, identically distributed variables sampled from a normal distribution with zero mean.B is the difference operator defined as follows

by Eq. (3.25). Where d is the order of the difference operator.

$$\Delta X_t = X_t - X_t - 1 = \Big(1 - B\Big) X_t \tag{3.25}$$

To fit the time series data with an ARIMA model following steps are involved.

- Scrutinize the time series data
  - Plot the data to find patterns and rregularities
  - If required, clean up any outliers or missing values
  - Take a logarithm of a series to help stabilize a strong growth trend
- Examine trends and seasonality and if present, remove trends and seasonality
- Use Augmented Dickey-Fuller test to check whether the time series is stationary or not. If the time series is not stationary then remove it by differencing the time-series.
- Use Auto-Correlation Function (ACF), and Partial Autocorrelation Function (PACF) plots to determine the order of differencing needed to choose order of the ARIMA model.
- Fit an ARIMA model
- Evaluate and iterate by checking the residuals, which have the same variance and are normally distributed. If there are visible patterns or bias, plot ACF and PACF. Refit model if needed. Compare model errors and fit using Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC).
- Calculate forecast using the chosen model.

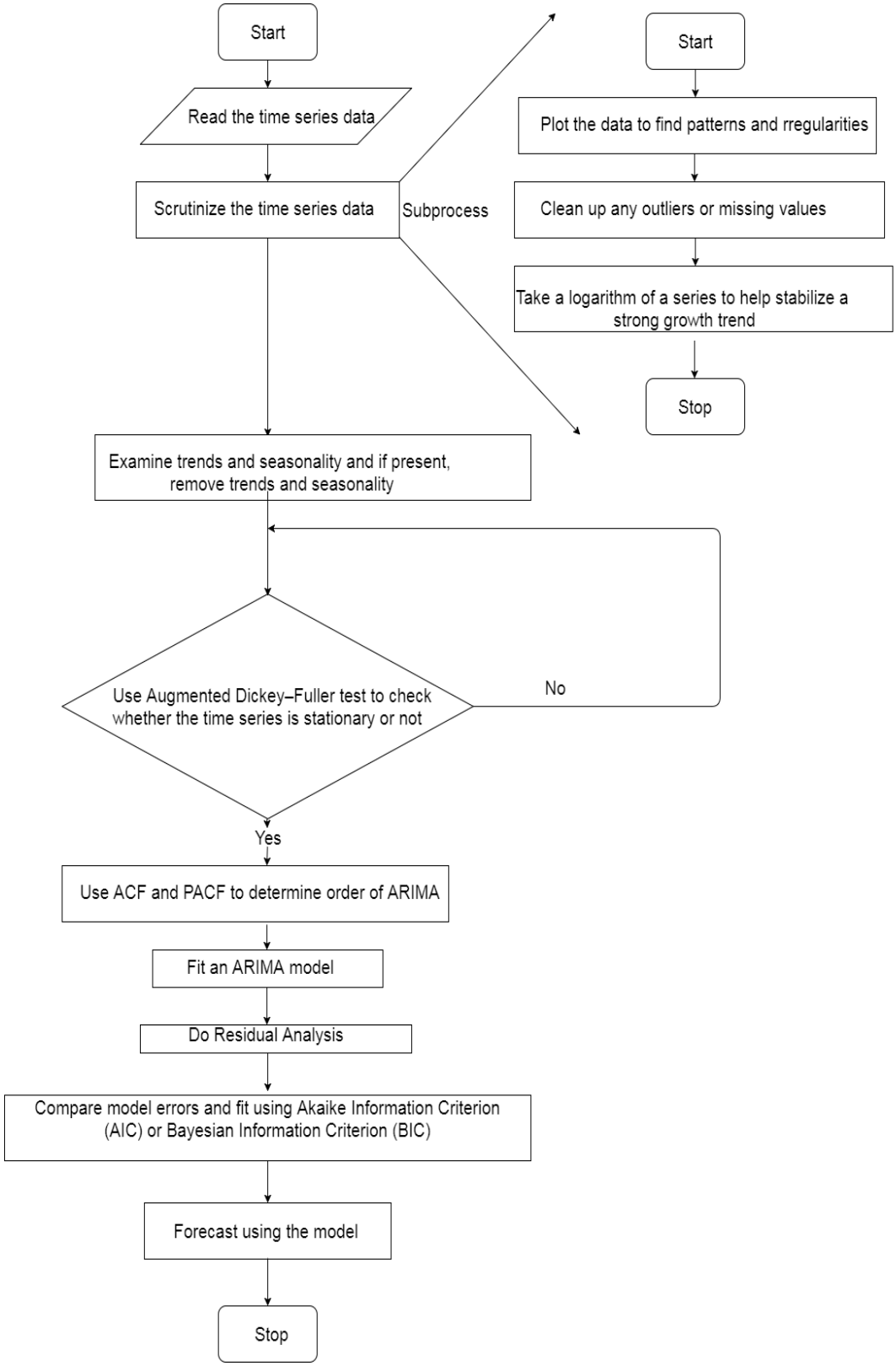

Figure 3.11: Flowchart to Find the Order of ARIMA Model

Figure 3.11 gives the flowchart of the method to find the order of ARIMA model.

3.2.3 Result Analysis of Resource Forecasting Using ARIMA

In Figure 3.12, we plotted free or available memory in Mbytes against time in hours. We used free memory in Mbytes for resource forecast throughout this work. We collected data for a time period of ten days with the time interval of half an hour. We considered the experimental setup explained in section 3.2.1.

The memory overcommit during this period is 4.11. Figure 3.13 illustrates the memory overcommit (15 minutes average) during this time. The memory overcommit is given by the Eq. (3.26). Where n is the number of VMs switched on. Swap usage in Mbytes, swap read Mbytes/sec and swap write Mbytes/sec are illustrated in Figures 3.14, 3.15, 3.16 respectively.

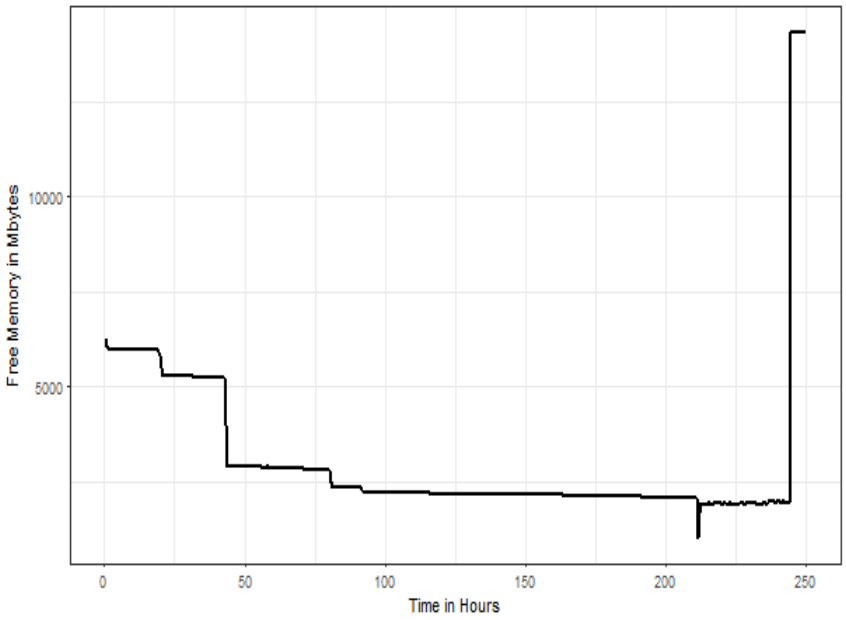

Figure 3.12: Free Memory in Mbytes Against Time in Hours

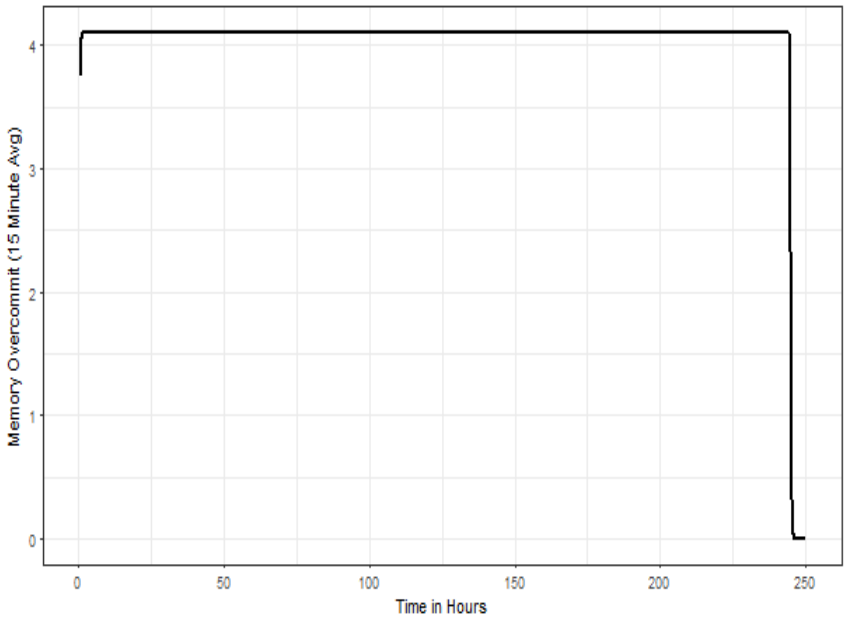

Figure 3.13: Memory Overcommit (15 minutes average) Against Time in Hours

$$Memory\ Overcommit = \frac{\sum_{i=0}^{n-1} Memory\ size\ of\ VMs}{Total\ memory\ available\ for\ Hypervisor} \tag{3.26}$$

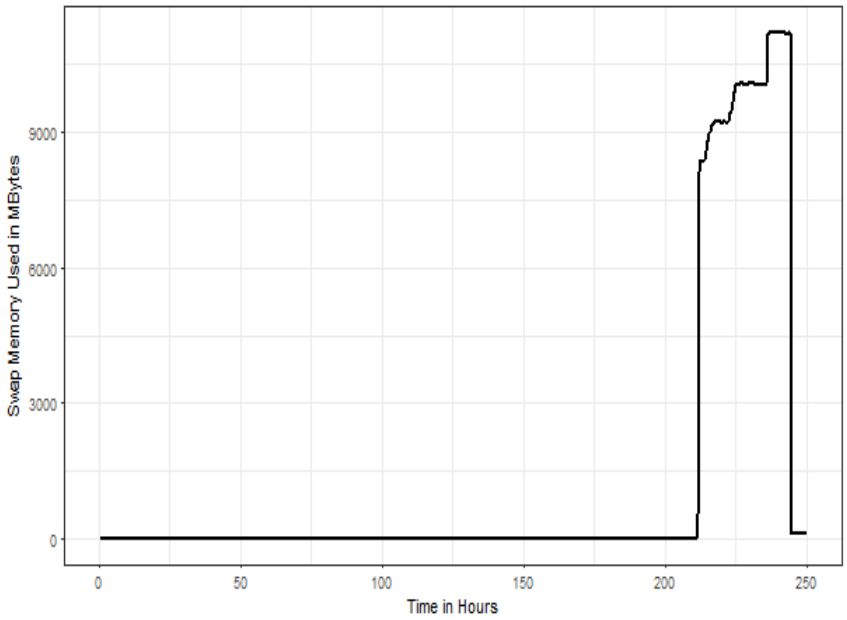

Figure 3.14: Swap Usage in Mbytes Against Time in Hours

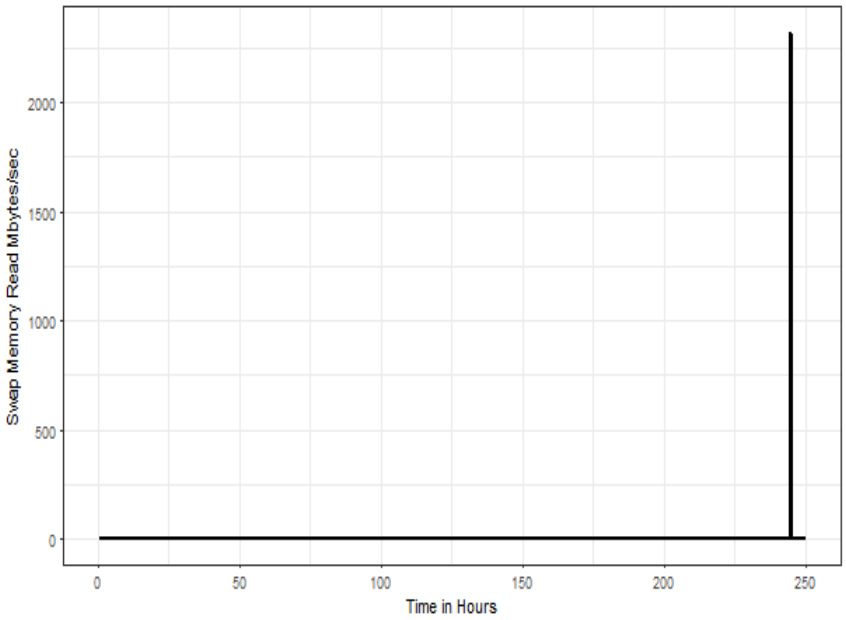

Figure 3.15: Swap Memory Read Mbytes/sec Against Time in Hours

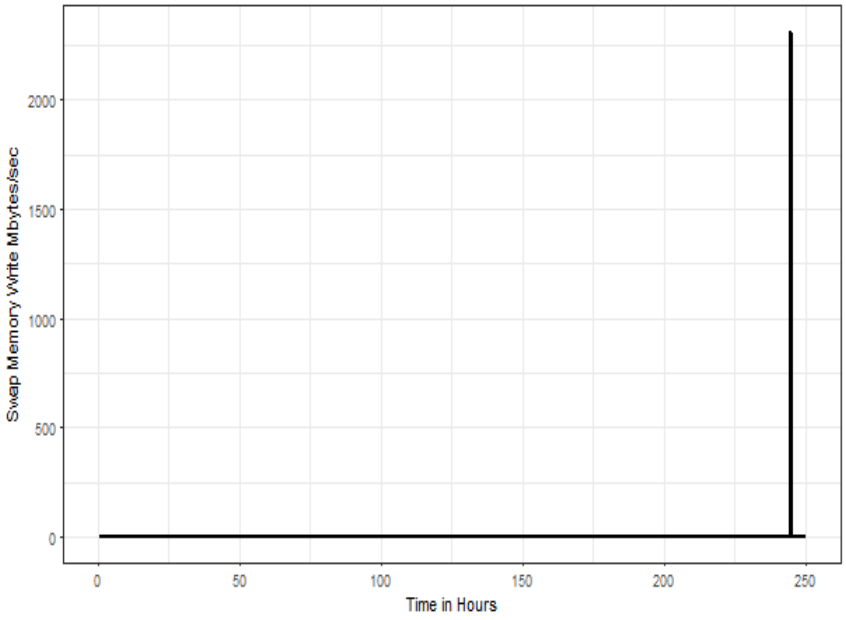

Figure 3.16: Swap Memory Write Mbytes/sec Against Time in Hours

It is clear from Figure 3.12 that at the 245^{th} hour the free memory has increased from 1900 Mbytes to 14000 Mbytes. From Figure 3.13, there is a clear indication that the Memory Overcommit (15 Minutes Average) has decreased from 4.11 to zero, which shows that some VMs or all VMs have been switched off and swapped into the secondary memory. Swap usage up to 211^{th} hour was nearly zero, and it

increased from 531 Mbytes at 212^{th} hour to 11182 Mbytes at 244^{th} hour apparently shows the evidence of thrashing. This is supported by the swap usage in Mbytes (Figure 3.14).

Figures 3.15, 3.16 swap read Mbytes per sec and swap write Mbytes per second up to 211^{th} hour was nearly zero, and it is increased from 1.11 Mbytes/sec at 212^{th} hour to 2324.5 Mbytes/sec at 244^{th} hour. This gives all reasons to believe that the hypervisor i.e. VMware ESXi 5.5.0 has used memory reclamation technique called hypervisor swapping to reclaim Memory. But that is not sufficient to stop the virtualized server consolidation system from an inevitable crash. So it is clear that system has reached a failure or failure probable state at 245^{th} hour. From the above discussion, it is evident that the free memory plot of Figure 3.12 shows the data has a huge variation after 211^{th} hour and it is due to the aging of the virtualized server consolidation system. Hence the data cannot be discarded as the outliers. Thus, we have taken the full dataset for the future analysis. Here, we were using the free memory collected against as the time series to model with ARIMA.

To find whether the free memory time series is stationary or not the autocorrelation of the series is plotted. The autocorrelation $\rho(k)$ between time series y_t and the k lagged version of y_t i.e. y_{t-k} is given by the Eq. (3.27).

$$\rho(k) = \frac{\frac{1}{n-k}\sum_{t=k+1}^{n}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(y_t - \bar{y})}\sqrt{\frac{1}{n-k}\sum_{t=k+1}^{n}(y_{t-k} - \bar{y})}} \tag{3.27}$$

Where $\bar{y}$ is the mean of y_t.

In time series analysis, the partial auto-correlation function (PACF) is used to identify the order of auto-regressive model. The use of this function was introduced as part of the Box-Jenkins approach to time series modeling, by plotting the partial autocorrelative functions one could determine the appropriate lags p in an $AR(p)$ model or an extended $ARIMA(p, d, q)$ model. Given a time series y_t, the partial auto-correlation of $lagk$, denoted $\phi(k)$, is the autocorrelation between y_t and y_{t-k} with the linear dependence of y_{t-1} to $y_{t-(k-1)}$ removed; this is obtained by solving Eq. (3.28).

Where $\rho(.)$ are the sample autocorrelations. This mapping between the sample autocorrelations and the partial autocorrelations is known as the Durbin-Levinson recursion (Durbin 1960).

$$\begin{bmatrix} \rho(0) & \rho(1) & \dots & \rho(k-1) \\ \rho(1) & \rho(0) & \dots & \rho(k-2) \\ \dots\dots\dots & & & \\ \rho(k-1) & \rho(k-2) & \dots & \rho(0) \end{bmatrix} \begin{bmatrix} \phi_{k1} \\ \phi_{k2} \\ \vdots \\ \phi_{kk} \end{bmatrix} = \begin{bmatrix} \rho(1) \\ \rho(2) \\ \vdots \\ \rho(k) \end{bmatrix} \tag{3.28}$$

To determine the order of differencing needed to make the time series stationary is one of the most important step. The correct order of differencing is the lowest number of differencing that yields a time series which fluctuates around a well-defined mean value and whose autocorrelation function (ACF) plot decays fairly rapidly to zero, either from above or below. If the autocorrelation function shows a slow decay, that means autocorrelations are positive out to a high number of lags (e.g., 10 or more), then it needs a higher order of differencing. Figures 3.17 and 3.18 show the ACF and PACF plots of free memory respectively. Figure 3.17, shows a slow decay in the ACF plot and this means the time-series is not stationary.

To reaffirm the nonstationarity, we conducted Augmented Dickey-Fuller (ADF) test on the time series. An augmented Dickey-Fuller test (ADF) checks whether a unit root is present or not in a time series sample. In this test, the null hypothesis checks whether the time series has a unit root or not. If the time series has a unit root, then the time series is nonstationary. From ADF, there is no statistical evidence that series is stationary. That means the time-series is nonstationary.

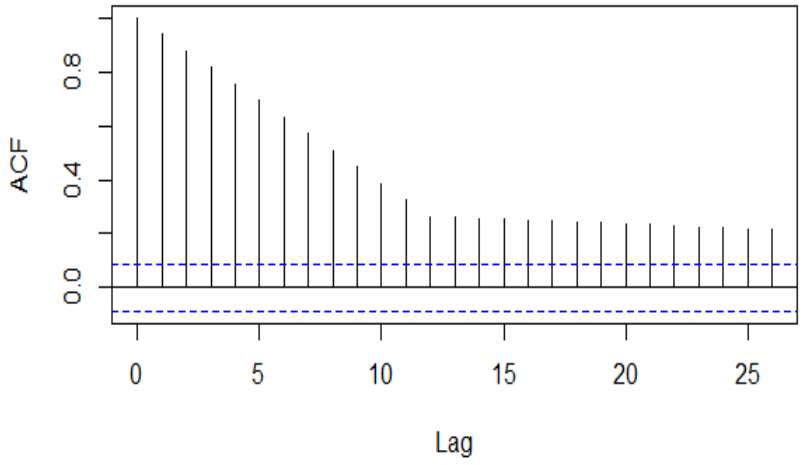

Figure 3.17: ACF of Free Memory

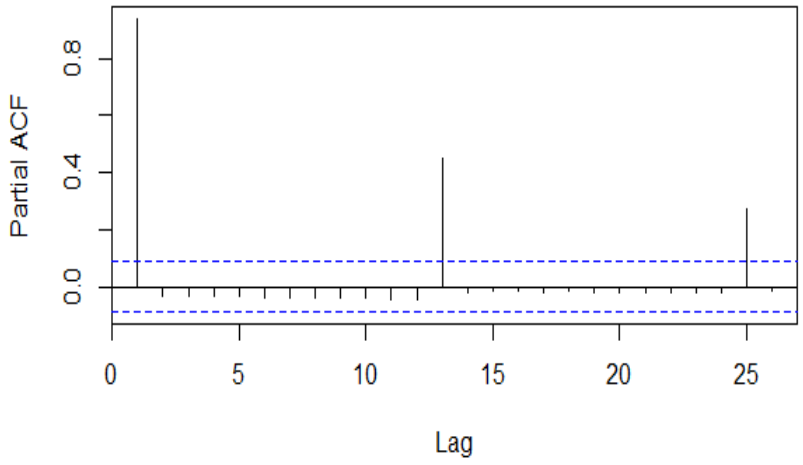

Figure 3.18: PACF of Free Memory

As the time series is non-stationary, we decided to difference the series. After differencing the time series, ADF test is conducted on the differenced time series, and it does not show any evidence of non-stationarity. Hence we decided to plot the ACF and PACF of the time series to determine the order of the ARIMA. Figures 3.19 and 3.20 show the ACF and PACF plots of the difference series.

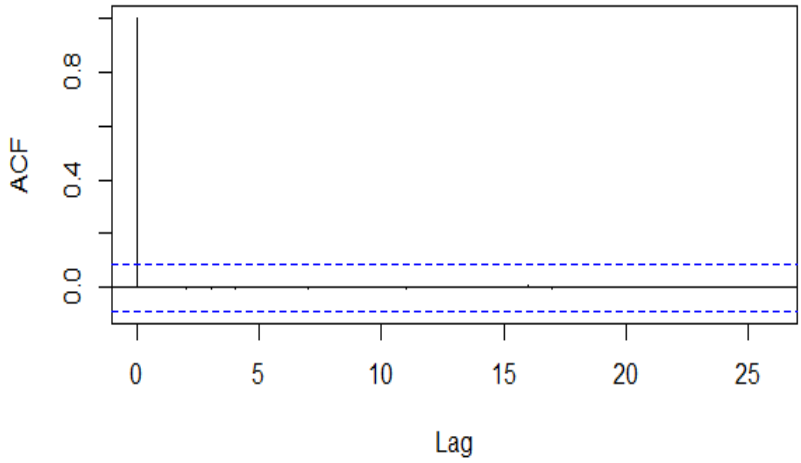

Figure 3.19: ACF of Differenced Time Series (Free Memory)

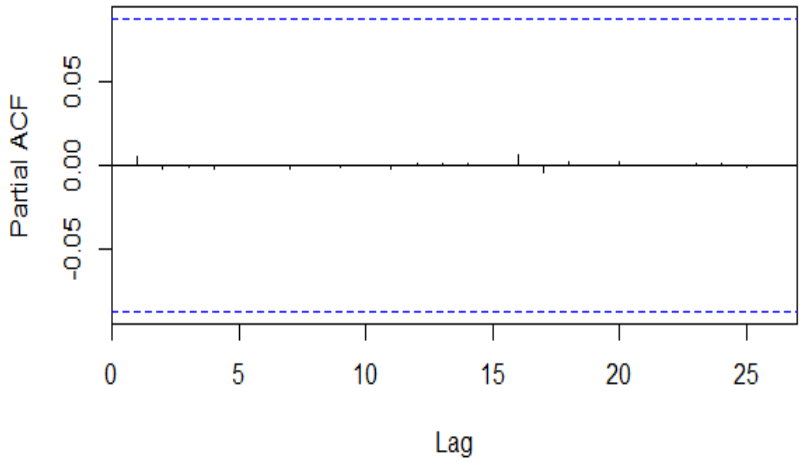

Figure 3.20: PACF of Differenced Time Series (Free Memory)

As there are no significant lags in both ACF (Figure 3.19) and PACF (Figure 3.20) plots of the differenced time series, we faced a difficulty in identifying the order AR (Auto-Regressive) and MA (Moving Average) terms in the ARIMA model. As the series is differenced once, we decided to check the goodness of fit of $ARIMA(0, 1, 0)$. We decided to check the goodness of fit of $ARIMA(1, 1, 0)$,

$ARIMA(0,1,1)$, and $ARIMA(1,1,1)$ as well. The basis of selection of $ARIMA(1,1,0)$, $ARIMA(0,1,1)$, and $ARIMA(1,1,1)$ is based on Ockham's razor (Ariew 1976). According to Ockham's razor, "Among competing hypotheses, the one with the fewest assumptions should be selected". To compare the models, we used Akaike information criterion (AIC), Bayesian information criterion (BIC) and Akaike information criterion corrected (AIC_c) values.

The Akaike information criterion (AIC) is a measure of the relative quality of a statistical model, for a given set of data. AIC provides a means for model selection. AIC deals with the trade-off between the goodness of fit of the model and the complexity of the model. It offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. AIC is given by Eq.(3.29)

$$AIC = 2k - 2.ln(L) \tag{3.29}$$

Where k is the number of parameters, and L is the maximum value of the likely hood function. Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. AIC_c is AIC with a correction for finite sample size given by the Eq. (3.30). Where n is the sample size.

$$AIC_c = AIC + \frac{2k(k+1)}{n-k+1} \tag{3.30}$$

Bayesian information criterion (BIC) is a goodness of fit criterion considered for model selection among a finite set of models. This is based on the likelihood function, and it is similar to the Akaike information criterion (AIC). When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in overfitting. Both BIC and AIC resolve this problem by introducing a penalty term for the number of parameters in the model; the penalty term is larger in BIC than in AIC. BIC is given by the Eq. (3.31).

$$BIC = -2ln(L) + k.ln(n) \tag{3.31}$$

Table 3.1 summarizes the comparison of ARIMA models: $ARIMA(0,1,0)$, $ARIMA(1,1,0)$,

$ARIMA(0,1,1)$, and $ARIMA(1,1,1)$. From Table 3.1, it is clear that none of the ARIMA models fits the free memory time series data well.

Table 3.1: Comparison of Different ARIMA models.

Model	AIC	AIC_c	BIC
$ARIMA(0,1,0)$	7738.29	7738.29	7742.5
$ARIMA(1,1,0)$	7740.27	7740.29	7748.69
$ARIMA(0,1,1)$	7740.27	7740.29	7748.69
$ARIMA(1,1,1)$	7742.27	7742.32	7754.91

Generally, a good forecasting method will produce residuals with the following properties:

- The residuals are uncorrelated. If there is correlation between residuals, then there is information left in the residuals which should be used in computing forecasts.
- The residuals have zero mean. If the residuals have a mean other than zero, then the forecasts are biased.
- The residual should be normally distributed.
- The residuals should have a constant variance.

We decided to carry out the residual analysis of the best fit amoung the chosen ARIMA models. Here we considered the $ARIMA(0,1,0)$ to analyse the residuals.

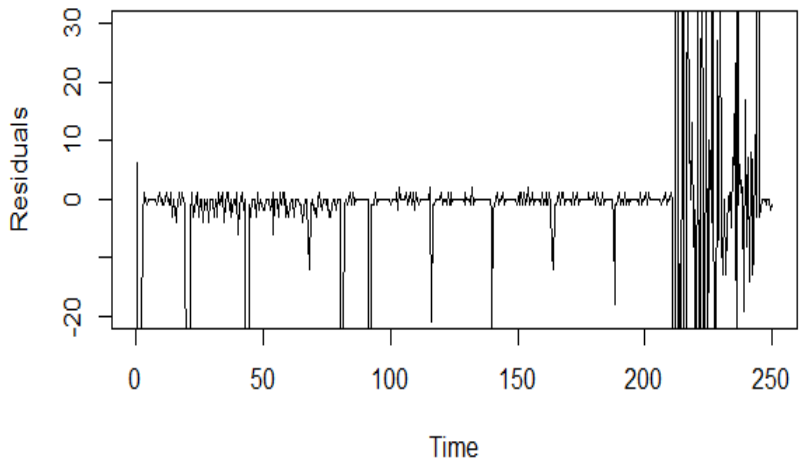

Figure 3.21: Residual Plot of the $ARIMA(0, 1, 0)$

The residual plot of the $ARIMA(0, 1, 0)$ is shown in Figure 3.21. It is evident from Figure 3.21; there is a huge difference in the variance of the residuals, especially at the end of the time series. That means the free memory shows a huge variation in usage pattern when the virtualized server consolidation system reaches a probable failure state. The mean of the residuals is not equal to zero; but its value is 16.18. To test the normality of the residual, quantile-quantile plot is plotted for the residuals of ARIMA(0,1,0) model. Figure 3.22 gives the quantile-quantile plot for the residuals of ARIMA(0,1,0) model, which shows the residuals are not normally distributed.

The residual analysis has explained the reasons for the inadequate fit of ARIMA models. It is evident from the model diagnostics that the time series is heteroscedastic in nature. In statistics, a collection of random variables is heteroscedastic if there are sub-populations that have different variabilities from others. In the next chapter, we discuss the variability of considered dataset i.e. the free memory, in detail.

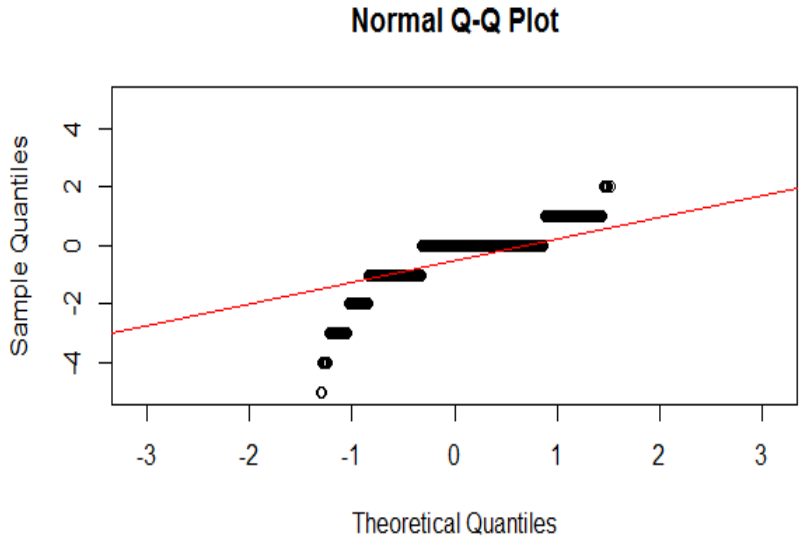

Figure 3.22: Quantile-Quantile Plot of the residuals of $ARIMA(0, 1, 0)$

3.3 Summary

This chapter considered the first two objectives of this research work, i.e. to establish aging effect in the resource usage data collected from server consolidation system. This chapter also examined the effect of aging on power usage and percentage CPU utilization when the virtualized server consolidation system moved to a likely failure state. Further, this chapter analyzed the forecasting errors associated with ARIMA models and thereby examining the presence of heteroscedasticity In the next chapter we will consider the third research objective, i.e. to understand the prediction errors associated with non-linear and heteroscedastic models and carry out the residual analysis for further possible improvement.

Chapter 4

Resource Consumption Analysis Using Non-Linear Models

This chapter focuses on heteroscedasticity present in the dataset, and we will use three non-linear models for resource forecasting, namely: Auto-Regressive Conditional Heteroscedasticity (ARCH), Generalized Auto-Regressive Conditional Heteroscedasticity (GARCH), and Artificial Neural Networks (ANN). In this chapter, first, we will give a brief introduction to heteroscedasticity then this chapter discusses the test for identifying the heteroscedasticity in the considered dataset followed by analysis of the dataset with three non-linear models as mentioned earlier.

4.1 Heteroscedasticity

A time series is said to be heteroscedastic if the sub-populations of the time series have different variabilities from others. Here "variability" is the variance or the statistical dispersion. Thus heteroscedasticity is the absence of homoscedasticity. The presence of heteroscedasticity in the dataset is a major worry for regression analysists because it can nullify many statistical tests of significance that presume the modeling errors are uncorrelated and uniform. The assumption of many linear regression models is that there is no heteroscedasticity in the dataset, breaking this may violate Gauss-Markov theorem (Gauss-Markov). The following are the Gauss-Markov assumptions about the error terms.

- They have mean zero: $\mathbb{E}[\varepsilon_i] = 0$.
- They have finite variance (Homoscedasticity): $\text{Var}(\varepsilon_i) = \sigma^2 < \infty$, and
- Distinct error terms are uncorrelated: $\text{Var}(\varepsilon_i) = \sigma^2 < \infty, \forall i \neq j$

where ε_i is the error term.

Breaking the assumptions of Gauss-Markov theorem in linear regression models may increase the variance of unbiased estimators, i.e., if the Gauss-Markov theorem is violated, then the Ordinary Least Square (OLS) estimators cannot be considered as the Best Linear Unbiased Estimators (BLUE). That means heteroscedasticity does not cause OLS coefficient estimates to be biased, but it can cause OLS estimates of the variance (and, thus, standard errors) of the coefficients are to be biased. This OLS estimates may vary possibly above or below the actual population variance, which in turn affect many statistical significance tests. Linear regression models using heteroscedastic data can still give an unbiased estimate of the relationship between dependent and independent variables, but inference made out of that may be wrong. A hypothesis test done on the coefficients of such estimates may produce type II error.

4.2 Tests of Heteroscedasticity

We employed three tests to identify whether heteroskedasticity exists in the time series or not. The first one is White test, second one is Golded-Quandt test and the third one is Breusch-Pagan test.

4.2.1 White Test

The White test (White 1980) checks whether the variance of the errors in a regression model is constant, i.e. homoskedasticity. Steps involved in White test are as follows:

- Regress dependent variable against independent variable using Ordinary Least Square (OLS) estimators

- Compute the OLS residuals, $\epsilon_1 \ldots \epsilon_n$
- Regress ϵ_i^2 against a constant, and all of the independent variables, the squares of the independent variables, and all possible interactions between the independent variables
- Compute coefficient of determination R^2 from the "auxiliary equation" (regression equation obtained by regressing ϵ_i^2 against a constant, and all of the independent variables, the squares of the independent variables, and all possible interactions between the independent variables).
- If the error term in the original model (model obtained by regressing dependent variable against independent variable) is homoscedastic (has a constant variance) then the coefficients in the auxiliary regression (besides the constant) should be statistically indistinguishable from zero, and the R^2 should be "small." Reject homoscedasticity if $nR^2 > \chi^2_{P-1,\alpha}$. Where P is the number of estimated parameters (in the auxiliary regression) and α is the significance level of the test.

We considered this test on the free memory which is obtained by doing the experiment as described in the section 3.2.1. We got $p - value < 0.05$, So we have rejected the null hypothesis, H_0 : Data is homoscedastic.

4.2.2 Goldfeld-Quandt Test

Goldfeld-Quandt divides the dataset into two parts then this test does separate regression analysis on these two parts and checks for homoscedasticity. Goldfeld-Quandt test computes Residual Sum of Squares (RSS) for both the regressions. If the ratio of the two RSS is close to one then the series is homoscedastic in nature.

Following steps are involved in performing the Goldfeld-Quandt test:

1. Rank the observations according to the variable value in the ascending order.

2. Omit c central observations (where c is specified apriori) and divide the remaining $(n-c)$ observations into two equal groups each containing $(\frac{(n-c)}{2})$ observations.

3. Fit separate linear regression to the first $(\frac{(n-c)}{2})$ observations and the second $(\frac{(n-c)}{2})$ observations using ordinary least squares (OLS) estimators.

4. Obtain theRSS_1 and RSS_2 and then compute the ratio as $GQ = \frac{RSS_1}{RSS2}$. If the ratio is close to one then the dataset is homoscedastic in nature.

Goldfeld-Quandt test statistics is nothing but the ratio of the residual sum of squares run on two different subsets of the sample. From the test results, it is found that Goldfeld-Quandt test statistics are greater than one, i.e; $GQ = 5.3286, df1 = 248, df2 = 248, p-value < 2.2e-16$. This shows the clear evidence of heteroscedasticity. In order to confirm the result Breusch-Pagan test was conducted.

4.2.3 Breusch-Pagan Test

It checks whether the variance of the errors from a regression is dependent on the values of the independent variables; If it depends, heteroscedasticity is present.

Let us consider a regression model, $y = \beta_0 + \beta_1 x$ where $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ is the OLS estimated model, where $\hat{u} = y - \hat{y}$ is the residual.

As per Gauss-Markov theorem, the residuals have a mean of zero, and it is uncorrelated with a constant variance. This test examines whether the Gauss-Markov assumption is valid or not, by fitting a simple model which is linearly related to independent variables. This model considers an auxiliary regression equation of the form $\hat{u}^2 = \gamma_0 + \gamma_1 x + v$ to regress the squared residuals to the independent variables.

Breusch-Pagan test is a chi-squared test. This test statistic has a distribution of $n\chi^2$ with k degrees of freedom. If the test statistic has a p-value below the significance level, then the null hypothesis of homoscedasticity is rejected.

The result of Breusch-Pagan test is $BP = 349.19, df = 1, p - value < 2.2e - 16$. Since the p-value is below the significance level (e.g. $p < 0.05$) hence the null hypothesis of homoscedasticity is rejected.

4.3 Resource Forecasting Using ARIMA-ARCH and ARIMA-GARCH

The major problem identified during the Resource forecasting using ARIMA model (section 3.2) is that the standardized error graph (Figure 3.21) of ARIMA fit showed volatility clustering when the virtualized server consolidation system started aging. From the graph, it is evident the residual of the ARIMA fit is not having constant variance. The residual violates the Gauss-Markov assumption, i.e., the residuals are normally distributed. But we could not find any significant correlation between the residuals. From the literature review, the none of the previous works try to address the volatility clustering in the system resource data. Here we decided to use the free physical memory of a virtualized server consolidation system (Figure 3.12) to fit Auto-Regressive Conditional Heteroscedasticity (ARCH) and Generalised Auto-Regressive Conditional Heteroscedasticity (GARCH) models.

The structure of the model can be described by the Eq. (4.1). In Eq. (4.1), the time series x_t is decomposed into a mean model $\mu_t(\theta)$ and a residual term ϵ_t.

$$x_t = \mu_t(\theta) + \epsilon_t \tag{4.1}$$

The mean model $\mu_t(\theta)$ may be an $ARIMA(p, d, q)$ where p is the order of the Auto-Regressive (AR), and q is the order of Moving Average (MA) terms. Where d is the order of difference. Here we considered an ARIMA as mean model. ϵ_t the residual may be represented by the Eq. (4.2).

$$\epsilon_t = \sigma_t z_t \tag{4.2}$$

Where, $\sigma_t^2(\theta) = E[(x_t - \mu_t(\theta))^2 | F_{(t-1)}], z_t\ N(0, 1)$. $\sigma_t^2(\theta)$ is the variance, and z_t is a white noice follows normal distribution with zero mean and unit variance. θ is a vector of unknown parameters. F_t is the set of information available at time t. This may include the present and past values of x_t, present and past values of the residuals or any other variable known at time t.

Therefore, the ARCH/GARCH modeling includes the following steps:

- Fit with a mean model, in our case it is an ARIMA model
- The residuals obtained from the above mean model is fitted with an ARCH/-GARCH model
- Compare the models with AIC, BIC, and AIC_c
- Do residual analysis on the best model
- Test the predictive power of model using Diebold-Mariano test

The overview of ARCH/GARCH modeling is illustrated in Figure 4.1.

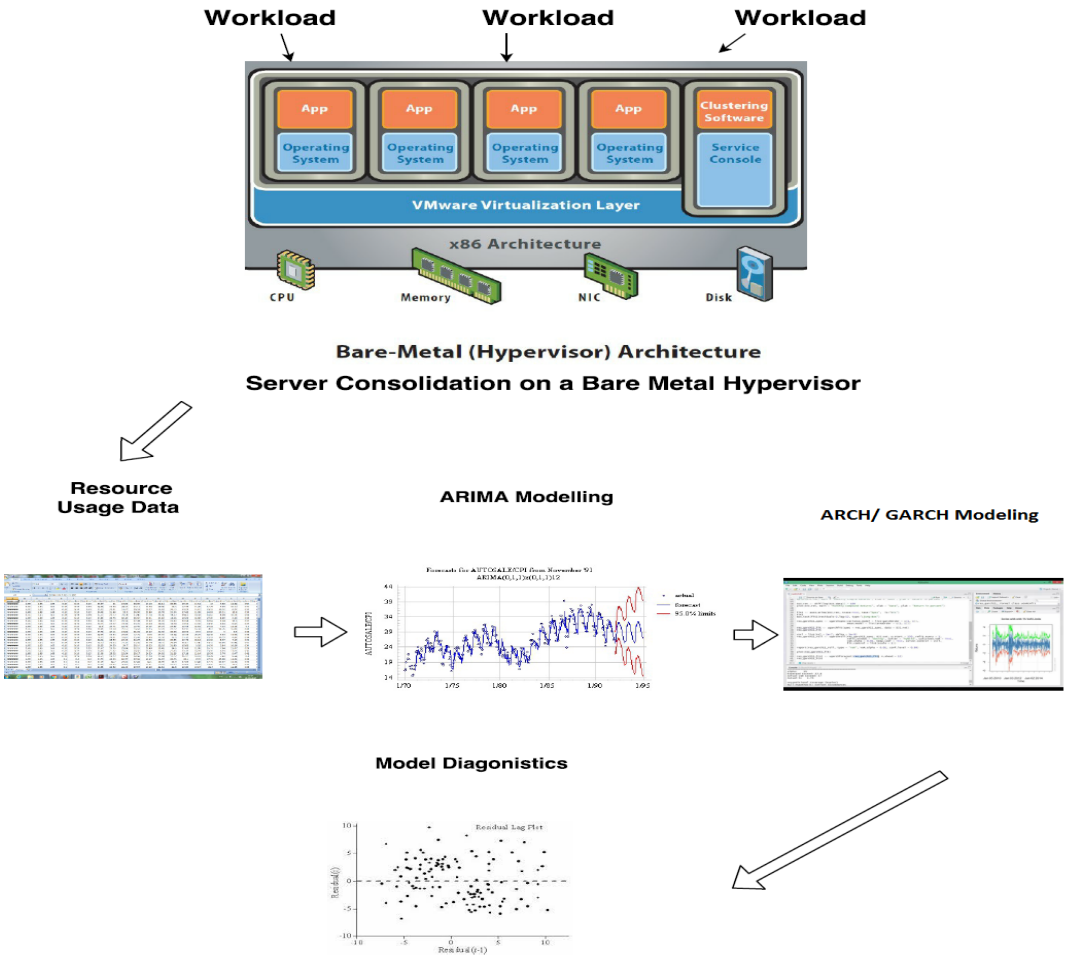

Figure 4.1: Overview of ARCH/GARCH modeling

4.3.1 Experimental Setup and Data Collection

The experimental setup is similar to that of section 3.1.1, here 24 Virtual Machines (VMs) were created on this server and installed with Ubuntu 14.04 operating system in all VMs. Apache 2.0 and PHP 5.0 were installed on these VMs. VMWare ESXi 5.5 was used as the hypervisor. Shell program is written to call the *test.php* page using *httperf* continuously with the rate of 400 requests per second. Server VMs used 1vCPU, 3GB RAM and 50GB hard disk with thin provisioning. 24VMs using 3GB RAM results in 72GB of logical RAM, but physically only 16 GB RAM is present. Hence main memory is over-committed to approximately 4.5 times. A memory leak is deliberately injected to accelerate the failure rate. System resources monitored were discussed in section 3.1.2. All memory reclamation techniques are enabled here except ballooning.

4.3.2 ARIMA Modeling

We have followed similar procedure that we have already expalined in the section 3.2.2. As the free memory showed signs of heteroscedasticity we have decided to model the residuals obtained from the best ARIMA fit as shown in Table 3.1 with ARCH and GARCH models, thus after integration we will obtain hybrid models, namely: ARIMA-ARCH and ARIMA-GARCH models.

4.3.3 ARCH/GARCH Modeling

Let us consider a regression model with a constant conditional variance given by, $Var(Y_t|X_{1,t}\ldots X_{p,t}) = \sigma^2$. Then the general form for the regression of Y_t on $X_{1,t}\ldots X_{p,t}$ is given by Eq. (4.3).

$$Y_t = f(X_{1,t}\ldots X_{p,t}) + \epsilon_t \tag{4.3}$$

Where Y is the dependent variable and X is independent variable. ϵ_t is independent of $X_{1,t}\ldots X_{p,t}$ and has an expectation of zero and constant conditional variance

of σ_ϵ^2. The function f is a conditional expection of Y_t given $X_{1,t} \dots X_{p,t}$. Here the conditional variance of Y_t is σ_ϵ^2.

Eq. (4.3) could be rewritten to allow conditional heteroskedasticity. Let the conditional variance of Y_t be $\sigma^2(X_{1,t} \dots X_{p,t})$. Then the regression model will be given by Eq. (4.4).

$$Y_t = f(X_{1,t} \dots X_{p,t}) + \sigma(X_{1,t} \dots X_{p,t})\epsilon_t \tag{4.4}$$

Autoregressive conditional heteroskedasticity (ARCH) is the condition where the variance of the current error term is a function of the previous error terms. The conditional variance is related to the squares of the previous error. So an $ARCH(p)$ assumes that the conditional variance $\sigma_t^2(\theta)$ is a linear function of the past p squared errors where θ is a vector of unknown parameters which could be estimated by maximum likelihood estimators. $ARCH(p)$ is given by Eq. (4.5).

$$\sigma_t^2(\theta) = \omega + \alpha_1\epsilon_{t-1}^2 + \dots + \alpha_p\epsilon_{t-p}^2 \tag{4.5}$$

The difference between ARCH and GARCH models is that in GARCH model time-dependent variance is a function of squared innovations and variance of previous time periods. The time-dependent variance of $GARCH(1,1)$ is given by Eq. (4.6).

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{4.6}$$

4.3.4 Result Analysis and Model Diagnostics

As mentioned earlier, the residuals of ARIMA fit of free memory of virtualized server consolidation system has a non-zero mean, and their residuals showed significant volatility clustering when the virtualized server consolidation system started aging, we decided to fit the residuals with ARCH/GARCH models. To find the best fit we considered the following procedure:

- Define a pool of candidate models,

- Estimate the models,
- Compare the estimated models based on the AIC, AIC_c and BIC
- Pick the best model for residual analysis.

To find the best fit we have to define a pool of candidate models. Since we considered four ARIMA models in the previous chapter, namely: $ARIMA(0,1,0)$, $ARIMA(1,1,0)$, $ARIMA(0,1,1)$, and $ARIMA(1,1,1)$., we decided to take the best two models among these four models, i.e. $ARIMA(0,1,0)$, and $ARIMA(1,1,0)$ as the mean model in this ARCH/GARCH fitting. Further, we decided to keep simple ARCH and GARCH models, namely: $ARCH(1)$, $ARCH(2)$, $GARCH(1,1)$, $GARCH(1,2)$, $GARCH(2,1)$, and $GARCH(2,2,)$, as these models have fewer parameters to estimate (Occam's Razor). According to Ockham's razor, "Among competing hypotheses, the one with the fewest assumptions should be selected." AIC, AIC_c, and BIC are used to select the best models out of this.

Table 4.1 summarizes the comparison of ARIMA +ARCH/GARCH models.

From Table 4.1, it is clear that none of the models fit the free memory time series data well. We have decided to use the best amoung ARIMA + ARCH/GARCH, say, $ARIMA(1,1,0)$+$GARCH(2,1)$ for residual analysis.

Generally, residuals of a good fit has the following properties:

- The residuals are uncorrelated.
- The residuals have zero mean.
- The residuals should be normally distributed.
- The residuals should have a constant variance.

To check whether the residuals of $ARIMA(1,1,0)$+$GARCH(2,1)$ are correlated or not, we have taken the ACF graph of the residuals. Figure 4.2 gives the ACF plot of $ARIMA(1,1,0)$+$GARCH(2,1)$ residuals.

Figure 4.2 shows no significant correlation between the residuals of $ARIMA(1,1,0)$+ $GARCH(2,1)$. Residuals have a non-zero mean. Figure 4.3 gives an indication that

Table 4.1: Comparison of Different ARIMA + ARCH/GARCH models.

Model	AIC	AIC_c	BIC
$ARIMA(0,1,0)$+$ARCH(1)$	7740.97	7740.99	7741.5
$ARIMA(0,1,0)$+$ARCH(2)$	7733.25	7733.56	7739.21
$ARIMA(0,1,0)$+$GARCH(1,1)$	7742.62	7742.62	7743.72
$ARIMA(0,1,0)$+$GARCH(1,2)$	7734.93	7735.12	7739.91
$ARIMA(0,1,0)$+$GARCH(2,1)$	7729.92	7729.9	7732.67
$ARIMA(0,1,0)$+$GARCH(2,2)$	7736.58	7736.58	7738.58
$ARIMA(1,1,0)$+$ARCH(1)$	7742.27	7742.32	7754.91
$ARIMA(1,1,0)$+ $ARCH(2)$,	7733.25	7733.25	7754.91
$ARIMA(1,1,0)$+$GARCH(1,1)$	7742.61	7742.71	7743.51
$ARIMA(1,1,0)$+$GARCH(1,2)$	7734.89	7734.89	7743.89
$ARIMA(1,1,0)$+$GARCH(2,1)$	7729.90	7729.91	7731.25
$ARIMA(1,1,0)$+$GARCH(2,2)$	7736.55	7736.55	7738.4

the residuals are not normal. Figure 4.4 gives the standardized residual graph of the $ARIMA(1,1,0)$+ $GARCH(2,1)$, shows sign of change in the variance of residuals when the virtualized server consolidation system moves to a failure stage.

4.3.5 Diebold-Mariano Test

To Compare the predictive accuracy of two forecasts from the best ARIMA+ARCH /GARCH model we used Diebold-Mariano Test (Giacomini and White 2006).

In Diebold-Mariano Test, we test the null hypothesis $H_0 : E(d_t) = 0 \ \forall t$ versus the alternative hypothesis $H_1 : E(d_t) \neq 0$.

The null hypothesis is observed when the two forecasts have the same accuracy. The alternative hypothesis is observed when the two forecasts have different levels of accuracy. We define the loss differential between the two forecasts by $d_t = g(e_{1t}) - g(e_{2t})$ and say that the two forecasts have equal accuracy if and only if the loss

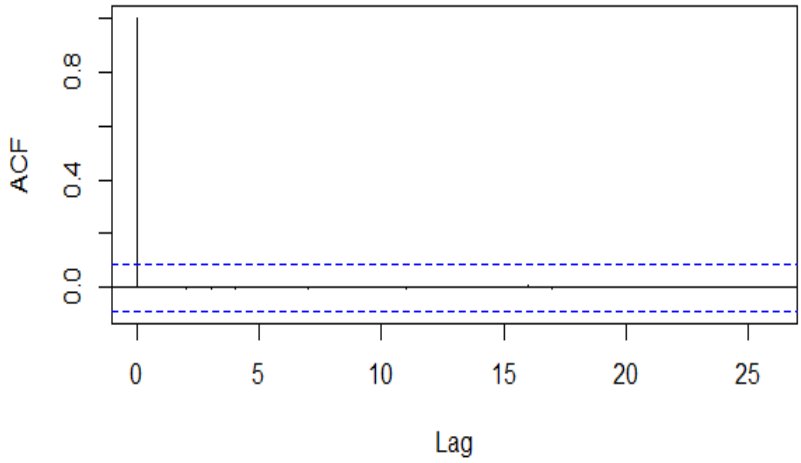

Figure 4.2: ACF graph of $ARIMA(1,1,0)$+$GARCH(2,1)$ residuals

differential has zero expectation for all t. The loss associated with forecast i is assumed to be a function of the forecast error, e_{it}, and is denoted by g(eit). $g(e_{it})$ is the square (squared-error loss) or the absolute value (absolute error loss) of e_{it}. The null hypothesis of zero difference will be rejected if the computed DM statistic falls outside the range of $-z_{\frac{\alpha}{2}}$ to $z_{\frac{\alpha}{2}}$. Here, the significance level of the test is $\alpha = 0.05$. Since this is a two-tailed test, hence 0.05 must be split such that 0.025 is in the upper tail and another 0.025 in the lower. The z-value that corresponds to -0.025 is -1.96, which is the lower critical z-value and 1.96 is the upper critical z-value. The null hypothesis of zero difference will be rejected if the computed DM statistic falls outside the range of -1.96 to 1.96.

Here we compared the $ARIMA(1,1,0)$+ $GARCH(2,1)$ and $ARIMA(0,1,0)$+ $GARCH(2,1)$ models, and obtained $DM = 0.17092$. We cannot reject the null hypothesis hence we conclude that both models have same predictive power. As the residuals show trends of nonlinearity and the goodness of fit of the ARIMA +ARCH/GARCH is moderate we decided to use ARIMA + ANN to fit the data.

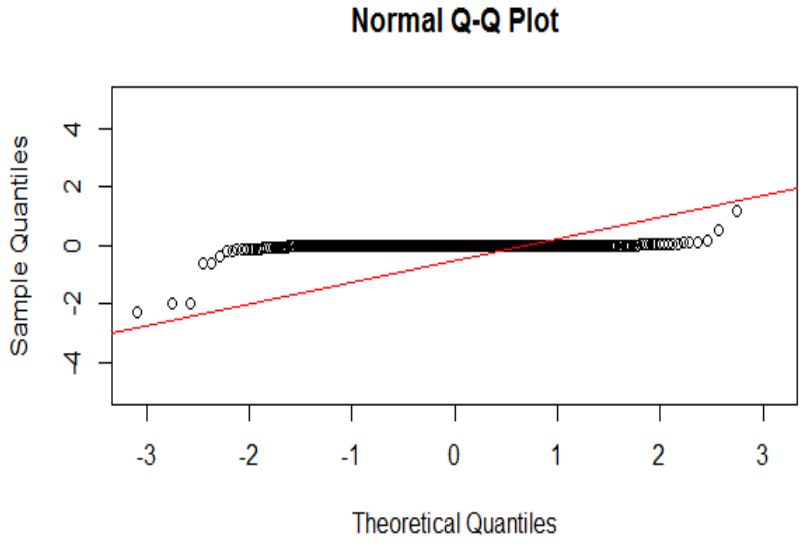

Figure 4.3: QQ plot of $ARIMA(1,1,0)$+$GARCH(2,1)$ residuals

4.4 Resource Forecasting Using ARIMA-ANN

The hybrid ARIMA-ANN modeling includes the following steps:

- Fit data with a linear model, in our case it is an ARIMA model
- The residuals obtained from the above mean model is fitted with an ANN model
- Compare the ANN models using mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE).

The overview of ARIMA-ANN modeling is illustrated in Figure 4.5.

4.4.1 Experimental Setup and Data Collection

The experimental setup is similar to that of section 3.1.1, here 24 Virtual Machines (VMs) were created on this server and installed with Ubuntu 14.04 operating system in all VMs. Apache 2.0 and PHP 5.0 were installed on these VMs. VMWare ESXi 5.5 was used as the hypervisor. Shell program is written to call the *test.php* page

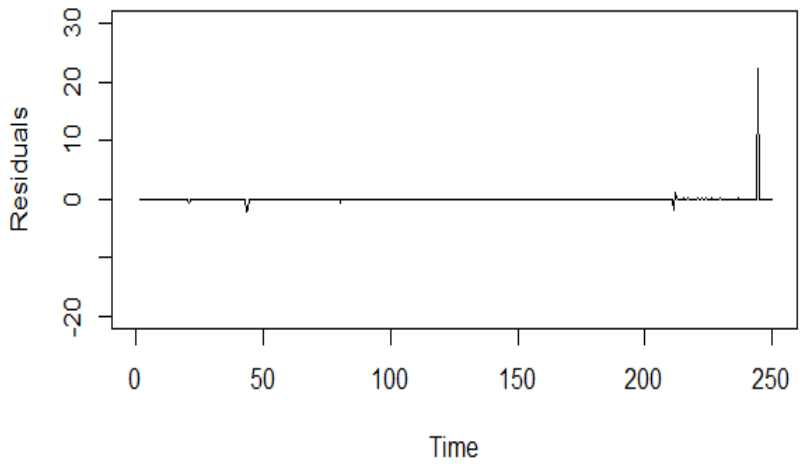

Figure 4.4: Residuals of $ARIMA(1,1,0)$+$GARCH(2,1)$

using *httperf* continuously with the rate of 400 requests per second. Server VMs used 1vCPU, 3GB RAM and 50GB hard disk with thin provisioning. The Client VM used 1GB RAM, one vCPU, 16GB of Hard disk. 24VMs using 3GB RAM results in 72GB of logical RAM, but physically only 16 GB RAM is present. Hence main memory is over-committed to approximately 4.5 times. A memory leak is deliberately injected to accelerate the failure rate. System resources monitored were discussed in section 3.1.2. All memory reclamation techniques are enabled here except ballooning.

4.4.2 ARIMA Modeling

We have followed similar procedure that we have expalined in the section 3.2.2. As the residuals free memory showed signs of non-linearity, we have decided to model the residuals obtained from the best ARIMA model shown in Table 3.1 with ANN model, thus it will become hybrid ARIMA-ANN model.

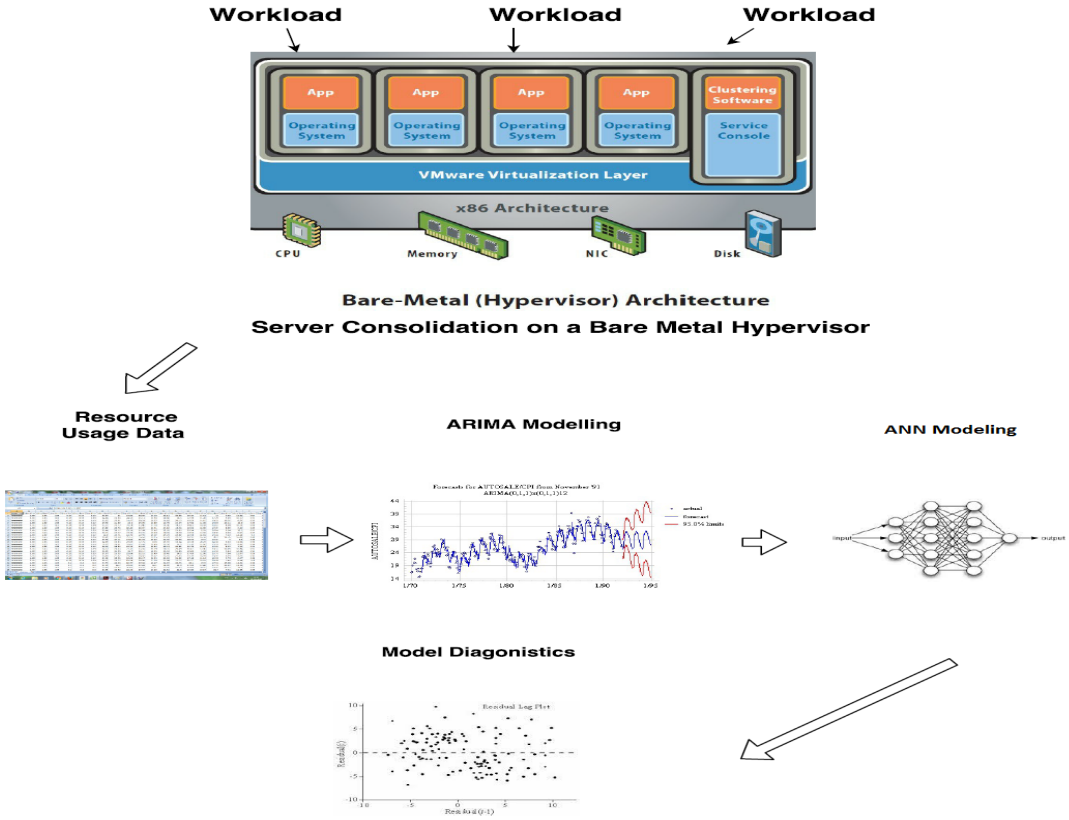

Figure 4.5: Overview of ARIMA-ANN modeling

4.4.3 ANN modeling

Artificial Neural Networks (ANNs) are computational model used in a large number of nonlinear problems. ANN process the data parallelly. The network architecture of ANNs consists of the input layer, hidden layer, and an output layer. A three-layer fully connected feedforward neural network is depicted in Figure 4.6.

The mathematical representation of relationship between the output y_t and the inputs $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ is given by Eq. (4.7).

$$y_t = \alpha_0 + \sum_{j=1}^{q} \alpha_j g(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-1}) + \epsilon_t \tag{4.7}$$

Where $\alpha_j (j = 1, 2, 3, \ldots, q)$ and $\beta_{ij} (i = 1, 2, 3, \ldots, p; j = 1, 2, 3, \ldots, q)$ are the model parameters, p is the number of input nodes, and q is the number of hidden nodes. The logistic function is often used as the hidden-layer transfer function, given

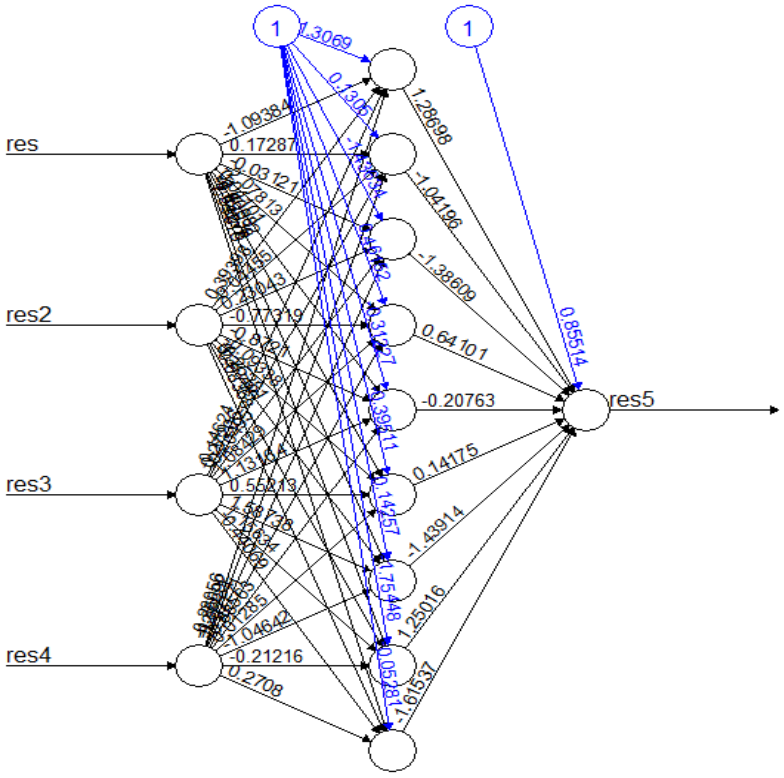

Figure 4.6: Feedforward ANN

by Eq. (4.8).

$$g(x) = \frac{1}{(1 + exp^{-x})} \tag{4.8}$$

In fact, the ANN model is a nonlinear functional mapping from the past observations $(y_{t-1}, y_{t-2}, \ldots, y_{t-p})$ to the future value (y_t), given by Eq. (4.9).

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \omega) + \epsilon_t \tag{4.9}$$

Where ω is a vector of all parameters, and $f(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \omega)$ is a function determined by the network structure, and connection weights and ϵ is the random error. The choice q of is data-dependent, and there is no systematic rule in deciding this parameter.

The number of input nodes in our case depends on the autocorrelation structure

of the time series. There is no hard and fast rule for selecting the number of p. Hence, a trial and error method is followed to select an appropriate p, as well as q.

4.4.4 ARIMA-ANN Hybrid Modeling

ARIMA and ANN models are good in linear and nonlinear domains, but individually they are not suitable for all circumstances. The approximation of ARIMA model for complex nonlinear problems may not yield good results. Similarly, ANN models will give mixed results in the linear domain. Since it is difficult to understand nature of the data in the real problem, we have considered a hybrid methodology that combines the capability of both linear and nonlinear models. Here, we combined ARIMA and ANN models.

Practically a time series may be considered to have a linear component and a non-linear component as shown in the Eq. (4.10).

$$y_t = l_t + n_t \tag{4.10}$$

Where l_t is the linear part and n_t is the nonlinear part. The two parts can be estimated separately from the data. First, the linear part is determined that is separated from the time series. The residuals of the linear part is fitted with the ANN model. This is the fundamental strategy behind this model. Let e_t be the residuals which are obtained by subtracting forecasted time series $\hat{l}_t$ from the original time series y_t. $\hat{l}_t$ is obtained by forecasting using ARIMA model in our case. e_t is obtained by the Eq. (4.11).

$$e_t = y_t - \hat{l}_t \tag{4.11}$$

If there are linear correlations left in residuals, then the linear models are not sufficient in forecasting the data. From the residual analysis, we could able to capture the nonlinear trends in the data. The nonlinear trends in the data could not be modeled with ARIMA. Here, we modeled the nonlinear trends available in

residuals with ANNs. If there are n input nodes, then the ANN model can be represented by the Eq. (4.12).

$$e_t = f(e_{t-1}, e_{t-2}, \ldots, e_{t-n}) + \epsilon_t \tag{4.12}$$

Where f is the non-linear function determined by ANN and ϵ is the random error. Here, we considered a hybrid system which consists of two steps. In the first step, an ARIMA model is used to analyze the linear part of the problem. In the second step, the residual from the ARIMA model is used to develop an ANN model.

The mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE) are used to evaluate the accuracy of forecasting. Eqs. (4.13), (4.14), and (4.15) give MAPE, MAE, and RMSE respectively.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{(A_t - F_t)}{A_t}\right| 100\% \tag{4.13}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n} A_t - F_t \tag{4.14}$$

$$RMSE = \sqrt{\left(\frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2\right)} \tag{4.15}$$

Where A_t is the actual value and F_t is the forecasted value and n is the number of data points considered.

4.4.5 Result Analysis and Model Diagnostics

As mentioned earlier (section 3.2.2) the residuals of ARIMA fit of free memory of virtualized server consolidation system has a non-zero mean, and their residuals showed significant volatility clustering when the virtualized server consolidation system started aging, we have decided to fit the residuals with ANN model. To find the best fit we considered the following procedure:

- Define a pool of candidate models,

- Estimate the models,
- Compare the estimated models based on $MAPE$, MAE and $RMSE$

To find the best fit we have to define a pool of candidate models. Since we considered four ARIMA models in the previous chapter, namely: $ARIMA(0,1,0)$, $ARIMA(1,1,0)$, $ARIMA(0,1,1)$, and $ARIMA(1,1,1)$; hence we decided to take the best two models among these four models, i.e. $ARIMA(0,1,0)$, and $ARIMA(1,1,0)$ as the linear model in this ARIMA-ANN hybrid fitting.

The residual plot of the $ARIMA(0,1,0)$ is shown in Figure 3.21. It is evident from Figure 3.21 that there is a huge difference in the variance of the residuals, especially at the end of the time series. That means the free memory shows a huge variation in usage pattern when the virtualized server consolidation system reaches a probable failure state. To test the normality of the residual, quantile-quantile plot is plotted for the residuals of ARIMA(0,1,0) model. Figure 3.22 gives the quantile-quantile plot for the residuals of ARIMA(0,1,0) model, which shows that the residuals are not normally distributed.

To check the traces of nonlinearity in the residual of $ARIMA(1,1,0)$, we decided to draw the standardized residual graph, quantile-quantile (q-q) plot (it is a graphical technique for determining if two data sets come from populations with a common distribution) of the $ARIMA(1,1,0)$ models. Figures 4.7 and 4.8 give the standardized residual graph, and quantile-quantile (q-q) plot of the $ARIMA(1,1,0)$ respectively.

Since traces of non-linearity is found in the residuals of both $ARIMA(0,1,0)$, and $ARIMA(1,1,0)$ models, hence artificial neural network (ANN) model is used to model the residuals obtained from forecasting by ARIMA. We considered three-layered feedforward neural network, trained by the resilient backpropagation algorithm (Riedmiller and Braun 1992). The hidden layer with 5, 10, and 15 artificial neurons has been used for finding the best model. We will refer the ANN model with five neurons, ten neurons, and fifteen neurons as ANN1, ANN2, and ANN3 respectively now onwards. Thus the pool of condidate models consists of the following models, namely: $ARIMA(0,1,0) + ANN1$, $ARIMA(0,1,0) + ANN2$,

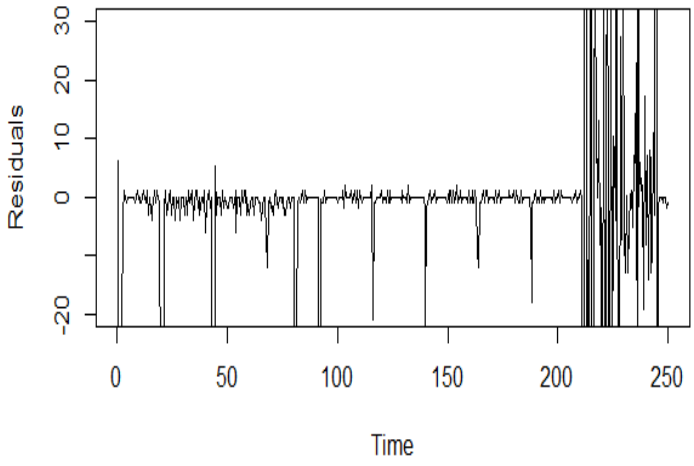

Figure 4.7: Residuals of $ARIMA(1,1,0)$

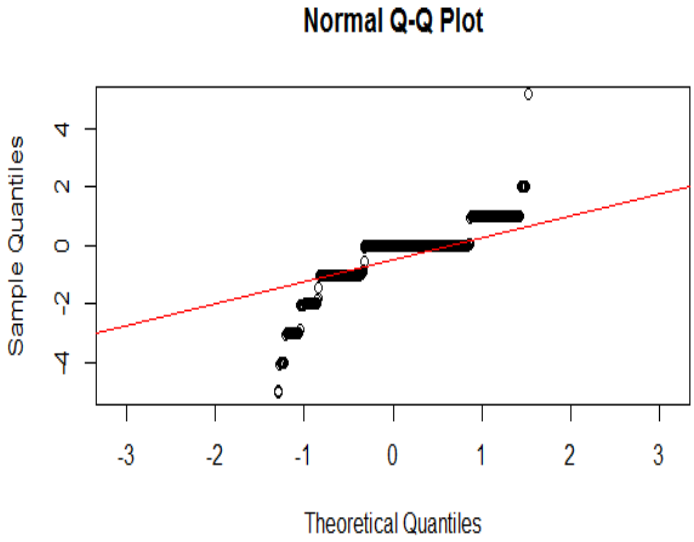

Figure 4.8: Q-Q Plot of Residuals-$ARIMA(1,1,0)$

$ARIMA(0,1,0)$ + $ANN3$, $ARIMA(1,1,0)$ + $ANN1$, $ARIMA(1,1,0)$+$ANN2$, and $ARIMA(1,1,0)$ + $ANN3$.

Table 4.2 summarizes the comparison of ARIMA + ANN models.

Table 4.2: Comparison of Different ARIMA + ANN Models.

Model	$MAPE$	MAE	$RMSE$
$ARIMA(0,1,0)$+$ANN1$	0.35	45	167
$ARIMA(0,1,0)$+$ANN2$	0.32	40	165
$ARIMA(0,1,0)$+$ANN3$	0.33	41	166
$ARIMA(1,1,0)$+$ANN1$	0.34	43	164
$ARIMA(1,1,0)$+ $ANN2$,	0.32	39	163
$ARIMA(1,1,0)$+$ANN3$	0.35	44	166

From Table 4.2, it is clear that the $ARIMA$ + ANN models fitted the considered data well. To check how $ARIMA$ + ANN model performs when compared to $ARIMA$ + $GARCH$ and $ARIMA$ models, we considered $MAPE$, MAE and $RMSE$ as the performance measures for comparison of the models, namely: $ARIMA(0,1,0)$, $ARIMA(1,1,0)$, $ARIMA(0,1,0)$ + $GARCH(2,1)$, $ARIMA(1,1,0)$ + $GARCH(2,1)$, $ARIMA(0,1,0)$ + $ANN2$, and $ARIMA(1,1,0)$ + $ANN2$. Table 4.3 summarizes the comparison of $ARIMA$, $ARIMA$ + $GARCH$, and $ARIMA$ + ANN models.

Table 4.3: Comparison of $ARIMA$, $ARIMA$ + $GARCH$, and $ARIMA$ + ANN.

Model	$MAPE$	MAE	$RMSE$
$ARIMA(0,1,0)$	0.88	40.14	562.18
$ARIMA(0,1,0)$+$ANN2$	0.32	40	165
$ARIMA(0,1,0)$+$GARCH(2,1)$	0.75	41	550.12
$ARIMA(1,1,0)$	0.89	40	562
$ARIMA(1,1,0)$+ $ANN2$,	0.32	39	163
$ARIMA(1,1,0)$+$GARCH(2,1)$	0.76	44	548.12

From Table 4.3, it is clear that the $ARIMA$ + ANN model fitted the considered

data well. The reason for the poor fit of the data by $ARIMA + GARCH$ and $ARIMA$ models may be due to the structural changes in the data. In the next chapter, we investigate whether the data contains structural breaks or not.

4.5 Summary

This chapter considered the third objective of this research work, i.e. to understand the prediction errors associated with non-linear and heteroscedastic models and carry out the residual analysis for further possible improvement. This chapter first checked the presence of heteroscedasticity in the data and further analyzed the goodness of fit of different $ARIMA + ARCH\ /\ GARCH$ models. Further, this chapter examined the forecasting errors associated with $ARIMA + ANN$ model. We checked the performance of $ARIMA + ANN$ when compared to $ARIMA + GARCH$ and $ARIMA$. We used $MAPE$, MAE and $RMSE$ as the measures of comparison. In the next chapter, we will consider the fourth objective, i.e. to investigate the presence of structural breaks and also to scrutinize the appropriateness of regime switching models for resource consumption analysis.

Chapter 5

Resource Consumption Analysis Using Regime Switching Models

In this chapter, we will investigate the presence of structural breaks and also scrutinize the appropriateness of regime switching models for resource consumption analysis. We decided to use two regime switching models, namely: Markov-Switching Generalized Auto-Regressive Conditional Heteroscedasticity (MS-GARCH) and Self-Exciting Threshold Auto-Regressive (SETAR), for fitting the dataset. Before we start analyzing the dataset with MS-GARCH and SETAR models, we decided to check the structural breaks in the dataset.

5.1 Structural Breaks

In statistics, a structural break, or structural change, is a huge difference in the moments of the dataset; such shift in the moments of the time series can lead to big forecasting errors. And this may affect the reliability of the model in general. In this research work, we considered two methods to identify the structural breaks in the data set, namely: 1) Chow Test 2) CUSUM Test.

To test the structural breaks in the dataset, we used the dataset from the experimental setup explained in the section 3.2.1. Figure 5.1 shows the time series plot of free memory in Mbytes against time in hours. We collected the data over a period of 10 days with the time interval of 30 minutes. We used free memory in Mbytes for resource forecast throughout this research work because the workload is more of memory intensive nature and time to exhaustion for such workload is smaller.

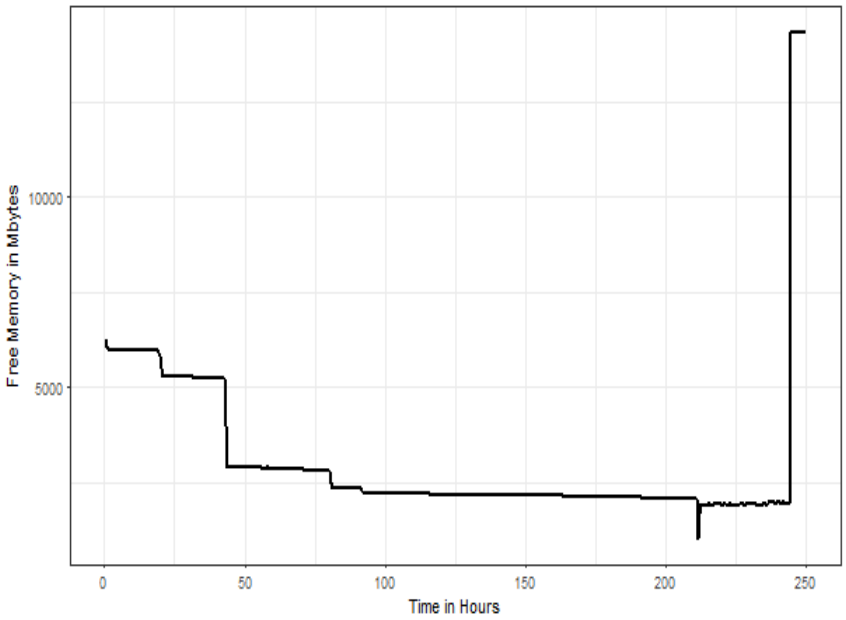

Figure 5.1: Free Memory in Mbytes Against Time in Hours

It is very evident from Figure 5.1, that there is a huge structural change at the end of the time series, this is the stage when the system started moving from the probable failure state to failure state. The reason for the structural change in the free memory is due to thrashing, and the details are explained in the section 3.2.2. We are interested in knowing whether there are any structural changes or not and further we are interested in checking the suitability of regime switching models for forecasting of resources in such cases.

5.1.1 Chow Test

The Chow test (Chow 1960) is used for testing whether the coefficients in two linear regressions on different data sets are equal or not. In statistics, this test is used to test for the presence of a structural break in time series analysis.

First, the time series data is split into two equal parts, then the coefficients of the two linear fits are compared to know whether a structural break is present or not. Let us consider a model for our data as defined by the Eq. (5.1).

$$y = c + \beta x + \epsilon \tag{5.1}$$

Where y is the dependent variable and x is the independent variable. Now we split the data into two groups; then the fit will be given by Eq. (5.2) and Eq. (5.3).

$$y_1 = c_1 + \beta_1 x_1 + \epsilon_1 \tag{5.2}$$

$$y_2 = c_2 + \beta_2 x_2 + \epsilon_2 \tag{5.3}$$

Now, we have to check whether the two linear regression fits are similar or not. Eq. (5.4) gives the null hypothesis and alternate hypothesis.

$$\begin{aligned} H_0 &: \beta_1 = \beta_2; c_1 = c_2 \\ H_1 &: \beta_1 \neq \beta_2; c_1 \neq c_2 \end{aligned} \tag{5.4}$$

Eq. (5.5) gives the Chow test statistic.

$$ChowStatistic = \frac{\frac{S_c - (S_1 + S_2)}{k}}{\frac{(S_1 + S_2)}{(N_1 + N_2 - 2k)}} \tag{5.5}$$

Where S_c is the sum of squared errors of prediction (SSE) from the combined data, S_1 is the SSE from the first group, S_2 is the SSE from the second group, N_1 and N_2 are the number of observations in each group and k is the total number of parameters. The Chow test statistics follow the F distribution with constant k and $N_1 + N_2 - 2k$ degrees of freedom.

Here, we used memory consumption data for 250 hours to test the presence of a structural break. We split the data into two equal parts, and the linear fit of the complete data along with the linear fits of the two equal parts which are depicted in Figure 5.2. The Chow statistic is calculated, and it has been used to analyze the presence of structural break. Since the calculated $F - value$ falls into the rejection region, hence we rejected the null hypothesis. Further it is also evident from Figure 5.2 that the regression lines do not have the same slope and intercept.

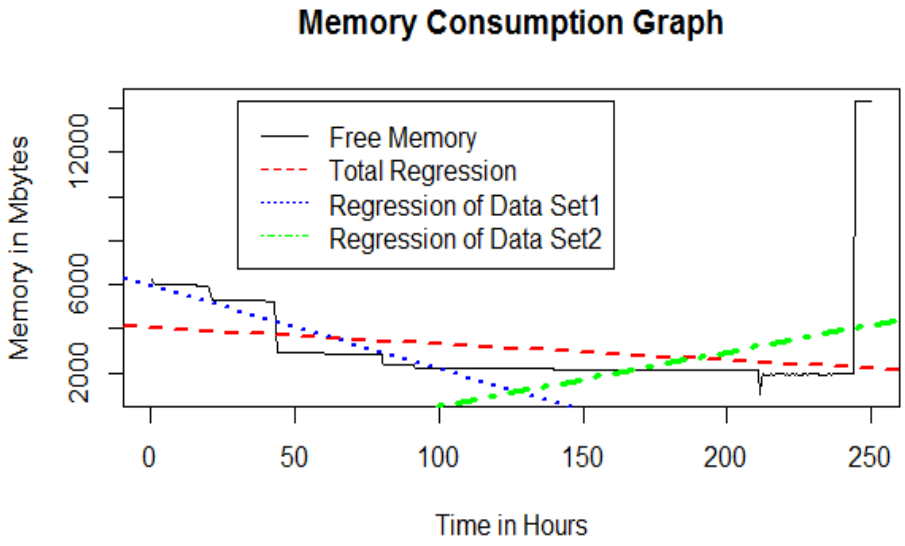

Figure 5.2: Chow Test

5.1.2 CUSUM Test

A CUSUM test (chart) is a control chart to monitor the change detection. CUSUM chart was initially devised in statistical quality control, to determine changes for deciding when to take corrective action. CUSUM chart displays the cumulative sums (CUSUMs) of the deviations of each sample value from the target value. The two-sided CUSUM chart uses a V-mask, instead of control limits, to determine when an out-of-control situation has occurred. The V-mask standardizes the deviations from the target value and plots the deviations from this value. Figure 5.3 shows the CUSUM chart of the considered time series and it shows that there is a structural change in the data.

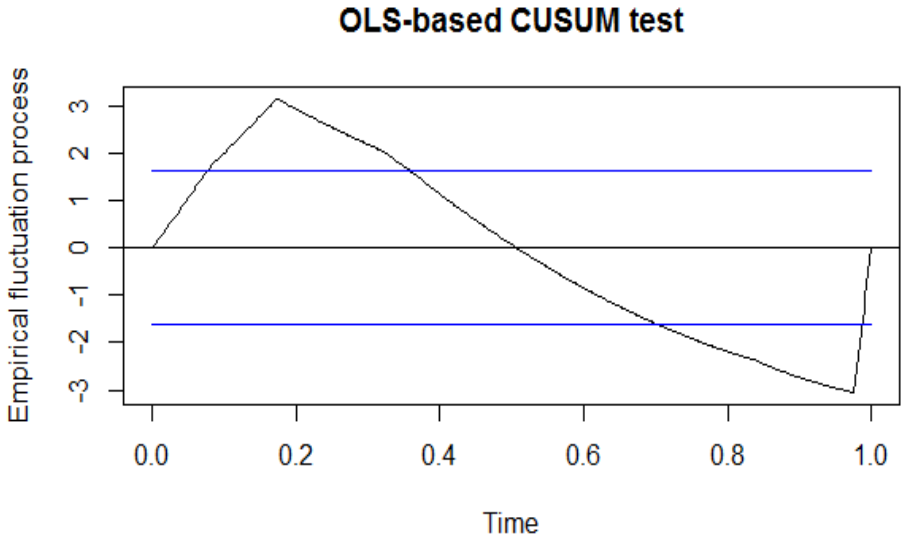

Figure 5.3: CUSUM Test

5.2 Resource Forecasting Using MS-GARCH

The MS-GARCH modeling includes the following steps:

- Collect the resource usage data.
- Fit the data with MS-GARCH model.
- Compare the MS-GARCH models using RMSE, MAE and MAPE values.

Figure 5.4 illustrates the overview of MS-GARCH modeling.

5.2.1 Experimental Setup and Data Collection

The experimental setup is similar to that of section 3.1.1, here 24 Virtual Machines (VMs) were created on this server and installed with Ubuntu 14.04 operating system in all VMs. Apache 2.0 and PHP 5.0 were installed on these VMs. VMWare ESXi 5.5 was used as the hypervisor. Shell program is written to call the *test.php* page using *httperf* continuously with the rate of 400 requests per second. Server VMs used 1vCPU, 3GB RAM and 50GB hard disk with thin provisioning. The Client

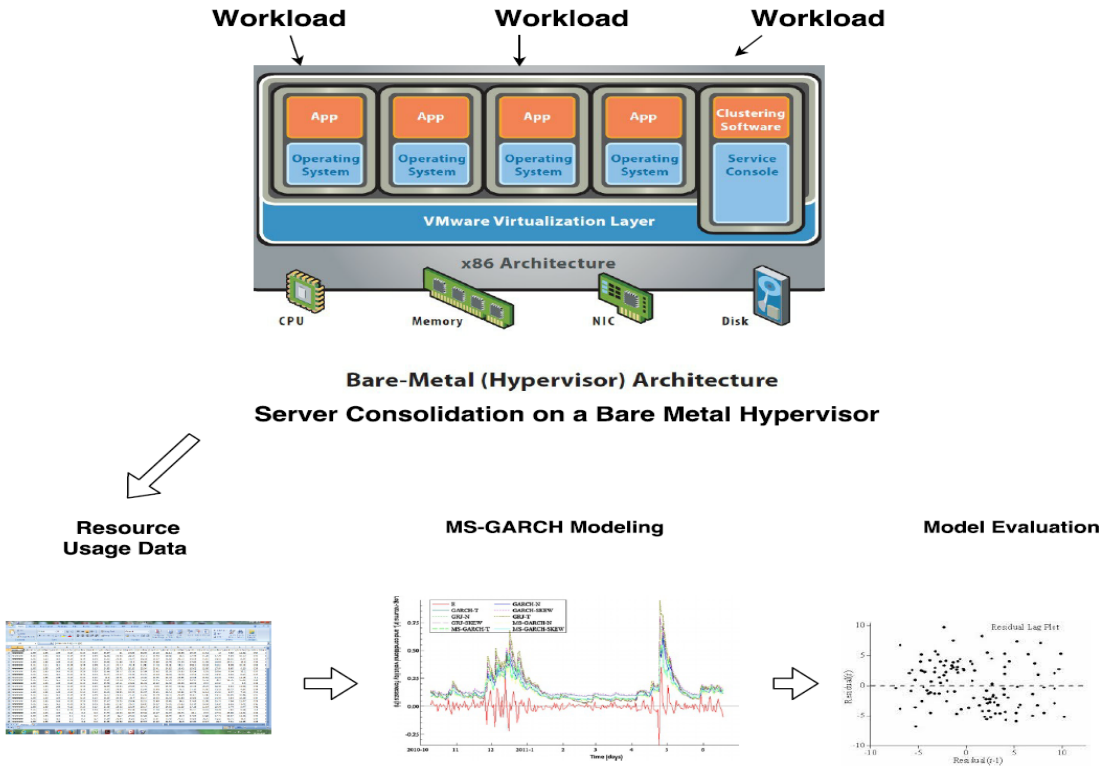

Figure 5.4: Overview of MS-GARCH Modeling

VM used 1GB RAM, one vCPU, 16GB of Hard disk. 24VMs using 3GB RAM results in 72GB of logical RAM, but physically only 16 GB RAM is present. Hence main memory is over-committed to approximately 4.5 times. A memory leak is deliberately injected to accelerate the failure rate. System resources monitored were discussed in section 3.1.2. All memory reclamation techniques are enabled in this experiment, except ballooning.

5.2.2 MS-GARCH Modeling

Regime switching models are used to model nonlinear trends in time series such as breaks, asymmetry or volatility clustering. The Markov regime switching model as proposed by Hamilton 1996 is one of the most popular models. In Markov regime switching model, the regime switching process is controlled by a first-order Markov chain, and thus it allows frequent changes at random points in time. The time

series is divided into different phases by this model. Each phase is specified with the different underlying stochastic process, and these phases are referred to as regimes or states.

Let s_t be a state variable which is a latent and unobservable in Markov regime switching model. For simplicity and further motivated by our data set, the present discussion in this chapter is limited to regime switching models with two states. The state space is defined by $S = 1, 2$. The state at time t is obtained from a two-state homogeneous first order Markov chain which is described by the transition probabilities p_{jj} for $j \in S$. The probability of being in the same state as in the previous period is given by the Eq. (5.6).

$$p_{jj} = Pr(s_t = j | s_{t-1} = j) \tag{5.6}$$

The transition matrix $\mathbf{P}$ is obtained by Eq. (5.7).

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} \\ \\ p_{21} & p_{22} \end{pmatrix} = \begin{pmatrix} p_{11} & 1 - p_{11} \\ \\ 1 - p_{22} & p_{22} \end{pmatrix} \tag{5.7}$$

Where $p_{ij} = Pr(s_t = j | s_{t-1} = i)$ denotes the probability of going from state i to state j. The current state depends only on the most recent state because of the Markov property. Hamilton 1996 used the Markov regime switching model that focusses on the mean behaviour of the variables. In this thesis, a stochastic process in state j, y_j may be specified by conditional variance models. The inference about the state varible is made from the observations of the stochastic process. Auto-Regressive Conditional Heteroscedasticity (ARCH) models are commonly used conditional variance models to characterize time series. They are used when the error terms have a characteristic variance. In ARCH models, the variance of the current error term or innovation is a function of previous time period's error terms. Often the variance is defined as the the squares of the previous innovations. The Generalized Auto-Regressive Conditional Heteroskedasticity Models (GARCH) assume

the variance of the current error term or innovation is a function of two parameters: squares of previous error terms and variance in the previous period.

$GARCH(1,1)$ can be defined by Eq. (5.8).

$$y_t = \mu_t + \sigma_t u_t \tag{5.8}$$

From Eq. (5.8), it is clear that the time series is decomposed into a conditional mean model and product of conditional variance and error term. μ_t may be an $ARIMA(p,d,q)$ where p is the order of the autoregressive terms, d is the order of differencing, and q is the order of moving average terms. The error term u_t is independent and identically distributed random variable with mean zero and unit variance. σ_t the conditional variance is given by Eq. (5.9) as follows,

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2 \tag{5.9}$$

Where ω, α, β are the parameters of the model and $\epsilon_t = y_t - \mu_t$. To ensure the positivity of conditional variance, the restriction $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$ is imposed. Markov-Switching Generalized Auto-Regressive Conditional Heteroskedasticity Models (MS-GARCH) model permits regime switching in the parameters. Let s_t be an ergodic Markov chain on a finite set $S = 1, \ldots, n$, with transition probabilities $p_{ij} = Pr(s_t = i | s_{t-1} = j)$. The MS-GARCH model is given by Eq. (5.10).

$$\begin{aligned} y_t &= \mu_{s_t} + \sigma_{s_t} u_t \\ \sigma_{s_t}^2 &= \omega_{s_t} + \alpha_{s_t}\epsilon_{t-1}^2 + \beta_{s_t}\sigma_{t-1}^2 \end{aligned} \tag{5.10}$$

Where ω_{s_t}, α_{s_t}, β_{s_t} are the parameters of the model and $\epsilon_t = y_t - \mu_{s_t}$. To ensure the positivity of conditional variance, the restriction $\omega_{s_t} > 0$, $\alpha_{s_t} \geq 0$, $\beta_{s_t} \geq 0$ is imposed.

Instead of GARCH in MS-GARCH model, we used Exponential Generalized Auto-Regressive Conditional Heteroskedastic (EGARCH) (Nelson and Cao 1992)

and Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) model (GLOSTEN et al. 1993). The difference between GARCH and EGARCH is only in the variance, is given by Eq. (5.11).

$$\begin{aligned} \log \sigma_t^2 &= \omega + \sum_{k=1}^{q} \beta_k g(Z_{t-k}) + \sum_{k=1}^{p} \alpha_k \log \sigma_{t-k}^2 \\ g(Z_t) &= \theta Z_t + \lambda(|Z_t| - E(|Z_t|)) \end{aligned} \tag{5.11}$$

Where σ_t^2 is the conditional variance, ω, β, α ,θ, and λ are coefficients. Z_t is a standard normal variable. $g(Z_t)$ allows the sign and the magnitude of Z_t to have separate effects on the volatility.

In GJR-GARCH model, variance is given by Eq. (5.12).

$$\sigma_t^2 = K + \delta\sigma_{t-1}^2 + \alpha\epsilon_{t-1}^2 + \phi\epsilon_{t-1}^2 I_{t-1} \tag{5.12}$$

Where K, δ, α, and ϕ are the parameters of the model, and $I_{t-1} = 0$ if $\epsilon_{t-1} \geq 0$, and $I_{t-1} = 1$ if $\epsilon_{t-1} < 0$.

5.2.3 Implementation of MS-GARCH Model

In this section, we will be explaining the implementation of MS-GARCH model, for simplicity, we limited our explanation to two regimes but the workflow described could be applied to n regimes. The workflow has been depicted in Figure 5.5.

The complete methodology of the training process and the forecast estimation has been logically divided into three phases. Following subsections comprehensively describe the functions in each phase. The outer loop explains that for each forecast the training happens again by including the forecasted value in the previous iteration. This is required because, with the newly added forecast value, the Markov regimes might change along with their probabilities of transition. This, in turn, results in different GARCH models built on each regime.

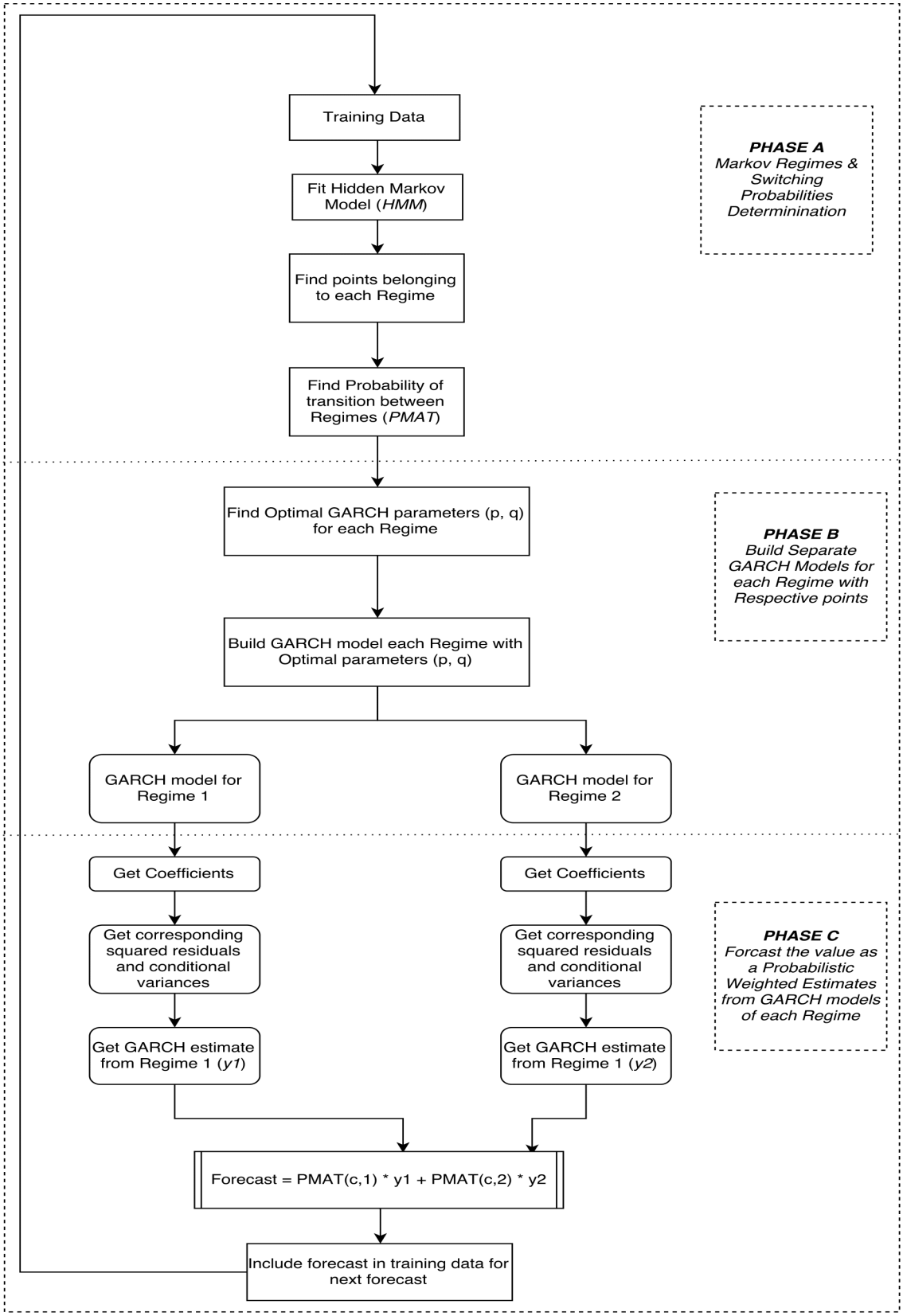

Figure 5.5: Workflow of MS-GARCH Modelling

The overall approach is that different Markov regimes are identified based on the training data by building a Hidden Markov Model (Ghahramani 2002), and the probability of transition between regimes is calculated. Then separate GARCH models are built on data points belonging to each regime. Then the resource forecast is made based on the current regime. For clarity the overall approach is divided into three phases: Phase A, Phase B and Phase C. In Phase A, Training data is taken, and Hidden Markov Model (HMM) is built on it. This provides probabilities of each data point belonging to different regimes. In Phase B, the parameter estimation and the order of different GARCH belonging to each regime are determined. Phase C forecasts the future values of time-series. RMSE, MAE and MAPE are estimated for the performance comparison.

Phase A:

In Phase A, the training data is fitted with a Hidden Markov Model (HMM), and in this thesis, we considered the free memory of virtualized server consolidation system. From this, we calculated the probabilities of each data point, and the probabilities will give the chances of data points belonging to different regimes. Thus, we knew which data points belong to which regime. Baum-Welch algorithm (Rabiner 1990) is used to find the unknown parameters of HMM and makes use of the forward-backward algorithm which is an inference algorithm for HMM which computes the posterior marginals of all hidden state variables given a sequence of observations. The algorithm makes use of the principle of dynamic programming to efficiently compute the values that are required to obtain the posterior marginal distributions in two passes. The first pass goes forward in time while the second pass goes backward in time; hence the name forward-backward algorithm.

The Viterbi-algorithm (Viterbi 2006) computes the most probable path of states for a sequence of observations for a given Hidden Markov Model. The Viterbi function returns which path each point belongs to. It returns a Viterbi path - A vector of strings, containing the most probable path of states. The Transition Probability Matrix (PMAT) is calculated from the path which contains the probabilities of

transition from one state to another state.

Phase B:

The order of the GARCH models of different regimes is obtained in Phase B. Different regimes will have a different order and set of parameters. The points belonging to different regimes could be obtained from HMM in Phase A. Based on the data points the order of GARCH model for each regime is decided. For selecting the order of the regimes, an iterative mechanism is used. Here Log-Likelihood ratio is used as the goodness of fit measure for selecting the optimal order of GARCH model for each regime.

Phase C:

In this phase, the future data is forecasted with MS-GARCH. The resource forecast should be done by taking the current state of the regime as the state of the latest data point. The probability of transition from the current regime to each of the regime is taken. The individual GARCH estimates of each regime are multiplied by the probability of transition from the current regime to that respective regime. This is essentially a probabilistic weighted average of the individual GARCH estimates of each Regime. Since, the data is highly volatile, which cannot be effectively predicted by a single GARCH model, hence we use an ensemble approach referred to as MS-GARCH, where estimates from multiple GARCH models are built on different subsamples of data depending on the nature of the data.

5.2.4 Results Analysis of Resource Forecasting with MS-GARCH

It is confirmed from section 5.1, that there is a structural change in the data, the next step is to fit the data with MS-GARCH model. The first step in Phase A is to fit a Hidden Markov Model (HMM). This provides probabilities of each data point belonging to different regimes. From this, we will be knowing which points belong to which regime. The probability plot for 2 states is shown in Figure 5.6. In this

Figure, the probabilities are rounded off to 1 and 0.

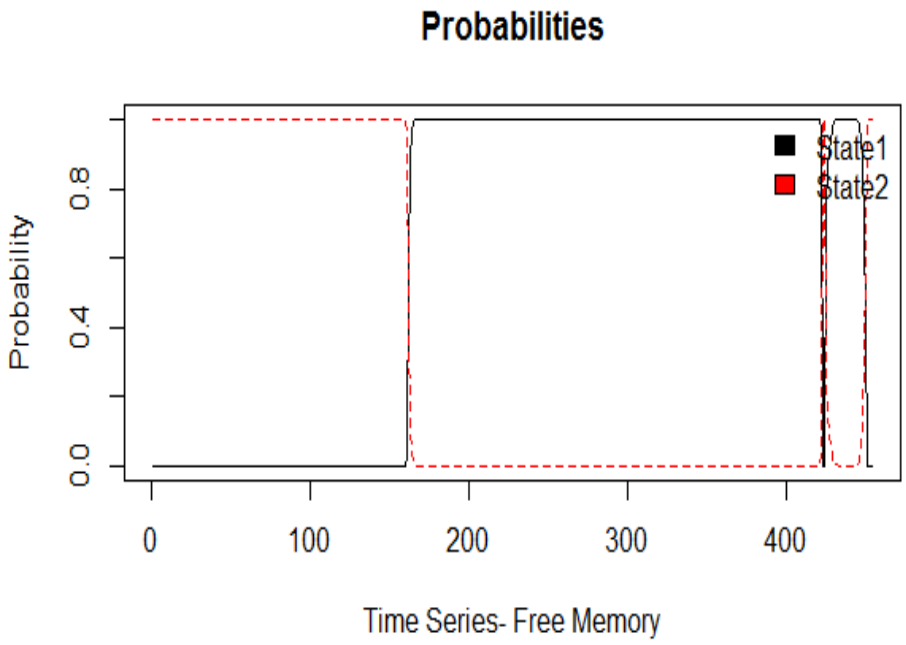

Figure 5.6: Probability Plot of Free Memory

Here we considered MS-GARCH model with two and three regimes, and then we estimated each model under three different distributional assumptions, namely the normal, student's t and the Generalized Error Distribution (GED). To compare models, we considered AIC, BIC and AIC_c values for all these models with error metrics $RMSE$, MAE and $MAPE$. For convenience, we named MS-GARCH models with two regimes as MS-GARCH-2REG and that with three regimes as MS-GARCH-3REG. Table 5.1 summarizes the comparison of MS-GARCH models.

Instead of GARCH in MS-GARCH, we used EGARCH and GJR-GARCH. Tables 5.2 and 5.3 summarize the comparison of MS-EGARCH and MS-GJR-GARCH models respectively.

To Compare the predictive accuracies of forecasts between MS-GARCH models we used Diebold-Mariano Test (Section 4.3.5). We considered MS-GARCH model with two regimes and normal distribution assumption as the base model to compare with rest of the models. Table 5.4 summarizes the comparison of predictive

Table 5.1: Comparison of MS-GARCH Models

Model	Distribution Assumption	AIC	AIC_c	BIC	$RMSE$	MAE	$MAPE$
MS-GARCH-2REG	Normal	567.25	568.19	575.56	43.12	26.43	0.23
MS-GARCH-3REG	Normal	667.15	668.23	685.21	48.6	27.9	0.26
MS-GARCH-2REG	Student's t	568.89	567.43	575.56	43.32	26.83	0.24
MS-GARCH-3REG	Student's t	667.25	668.23	685.61	48.21	27.91	0.26
MS-GARCH-2REG	GED	568.75	568.34	575.56	43.12	26.43	0.24
MS-GARCH-3REG	GED	668.25	669.23	687.56	49.44	28.32	0.27

Table 5.2: Comparison of MS-EGARCH Models

Model	Distribution Assumption	AIC	AIC_c	BIC	$RMSE$	MAE	$MAPE$
MS-EGARCH-2REG	Normal	568.31	568.79	574.42	44.09	27.33	0.25
MS-EGARCH-3REG	Normal	689.21	696.98	698.21	49.36	27.01	0.34
MS-EGARCH-2REG	Student's t	574.21	576.01	595.56	46.32	25.94	0.29
MS-EGARCH-3REG	Student's t	676.25	675.94	685.32	48.21	27.91	0.35
MS-EGARCH-2REG	GED	579.75	582.34	591.06	43.12	25.93	0.31
MS-EGARCH-3REG	GED	675.25	671.03	689.19	49.44	29.15	0.32

Table 5.3: Comparison of MS-GJR-GARCH Models

Model	Distribution Assumption	AIC	AIC_c	BIC	$RMSE$	MAE	$MAPE$
MS-GJR-GARCH-2REG	Normal	562.11	565.99	564.42	42.07	26.33	0.24
MS-GJR-GARCH-3REG	Normal	699.22	699.98	698.71	49.10	27.01	0.24
MS-GJR-GARCH-2REG	Student's t	574.21	576.01	595.56	46.32	25.94	0.30
MS-GJR-GARCH-3REG	Student's t	676.25	675.94	685.32	48.21	27.91	0.35
MS-GJR-GARCH-2REG	GED	582.75	587.34	591.06	46.72	25.93	0.31
MS-GJR-GARCH-3REG	GED	678.51	679.72	699.19	50.44	29.15	0.36

Table 5.4: Diebold-Mariano Test Results

Model	Distribution Assumption	DM	$p-value$
MS-GARCH-3REG	Normal	0.3332	0.7020
MS-GARCH-2REG	Student's t	0.2757	0.7799
MS-GARCH-3REG	Student's t	0.1702	0.8644
MS-GARCH-2REG	GED	0.3422	0.5621
MS-GARCH-3REG	GED	1.0292	0.6921
MS-EGARCH-2REG	Normal	1.1082	0.6321
MS-EGARCH-3REG	Normal	0.2392	0.1286
MS-EGARCH-2REG	Student's t	0.7402	0.2407
MS-EGARCH-3REG	Student's t	0.3414	0.9036
MS-EGARCH-2REG	GED	0.3206	0.4798
MS-EGARCH-3REG	GED	0.1985	0.2336
MS-GJR-GARCH-2REG	Normal	0.4298	0.6898
MS-GJR-GARCH-3REG	Normal	-0.7411	0.6099
MS-GJR-GARCH-2REG	Student's t	0.2272	0.4543
MS-GJR-GARCH-3REG	Student's t	0.1633	0.6012
MS-GJR-GARCH-2REG	GED	0.3304	0.7056
MS-GJR-GARCH-3REG	GED	0.1633	0.8644

accuracies using Diebold-Mariano Test.

Here, the significance level of the test is $\alpha = 0.05$. This is a two-tailed test with significance level of 0.05. The z-value that corresponds to -0.025 is -1.96, which is the lower critical z-value and 1.96 is the upper critical z-value. The null hypothesis of zero difference will be rejected if the computed DM statistic falls outside the range of -1.96 to 1.96. From Table 5.4, we cannot reject the null hypothesis so we can conclude that all models have relatively same predictive power.

5.3 Resource Forecasting Using SETAR

The next model which we considered in this thesis contribution is Self-Exciting Threshold Auto-Regressive (SETAR) (Tong and Lim 1980) model. Self-Exciting Threshold AutoRegressive (SETAR) models are generally applied to time series data as an extension of autoregressive models; this is to grant a higher degree of flexibility in model parameters through a regime switching behavior. Threshold Auto-Regressive (TAR) models are the popular class of nonlinear time series models which are simple and easy to understand. These models are capable of forecasting time series with complex nonlinear dynamics.

SETAR model is a time series model used for forecasting future values. SETAR is a regime switching model, considered when the time series shows significant structural breaks. The switch from one regime to another regime depends on the past values of the time series (hence the Self-Exciting portion of the name). The model is usually referred to as the $SETAR(k, p)$ model where k is the number of regimes and p is the order of the autoregressive part. Since these can differ between regimes, the p portion is sometimes dropped, and hence these models are denoted simply as $SETAR(k)$.

The SETAR modeling includes the following steps:

- Collect the resource usage data.
- Fit the data with SETAR model.
- Compare the SETAR models using RMSE, MAE and MAPE values.

Figure 5.7 illustrates the overview of SETAR modeling.

5.3.1 Experimental Setup and Data Collection

The experimental setup is similar to that of section 3.1.1, here 24 Virtual Machines (VMs) were created on this server and installed with Ubuntu 14.04 operating system in all VMs. Apache 2.0 and PHP 5.0 were installed on these VMs. VMWare ESXi

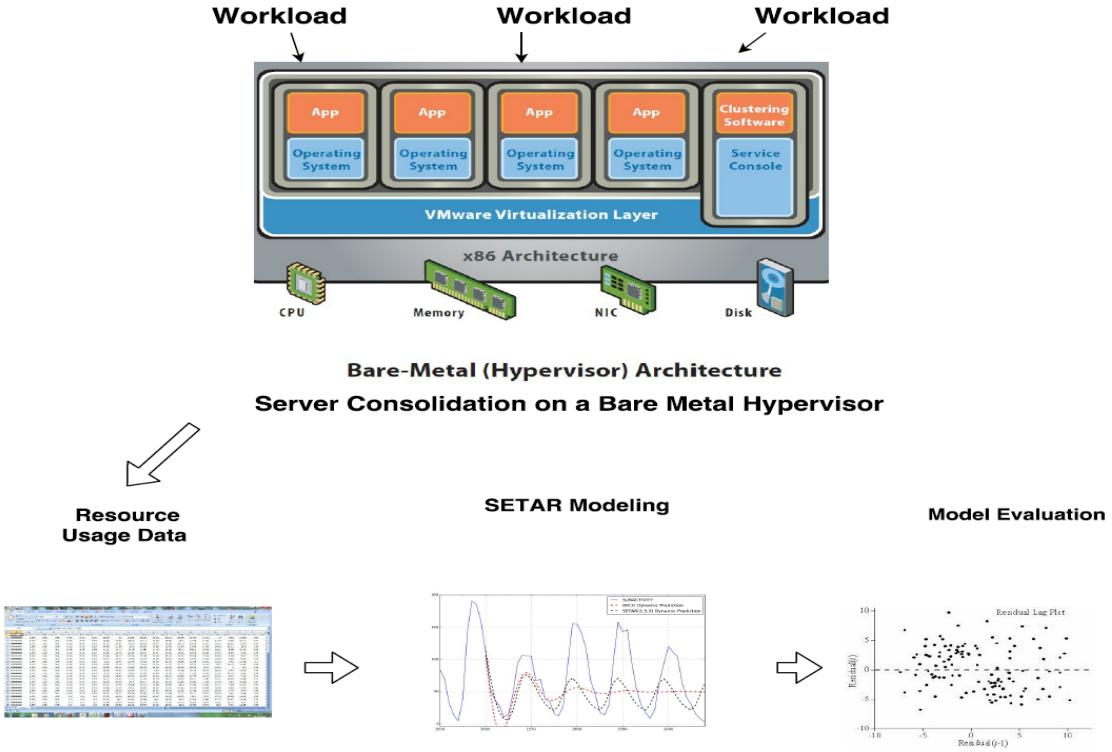

Figure 5.7: Overview of SETAR Modeling

5.5 was used as the hypervisor. Shell program is written to call the *test.php* page using *httperf* continuously with the rate of 400 requests per second. Server VMs used 1vCPU, 3GB RAM and 50GB hard disk with thin provisioning. The Client VM used 1GB RAM, one vCPU, 16GB of Hard disk. 24VMs using 3GB RAM results in 72GB of logical RAM, but physically only 16 GB RAM is present. Hence the main memory is over-committed to approximately 4.5 times. A memory leak is deliberately injected to accelerate the failure rate. System resources monitored were discussed in section 3.1.2. All memory reclamation techniques are enabled in this experiment, except ballooning.

5.3.2 SETAR

Consider a simple $AR(p)$ model as shown in Eq.(5.13), for a time series $y(t)$.

$$y(t) = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \sigma\epsilon_t \tag{5.13}$$

Where $\phi_i (i = 1, 2, ..p)$ is the AR coefficients, $\epsilon_t \sim iidN(0, \sigma^2)$ where σ^2 is constant variance. The model parameters, $\phi_i (i = 1, 2, ..p)$ and σ^2 are constants with time t. TAR allows the model parameters to change according to the value of a weakly exogenous threshold variable z_t for capturing nonlinear trends.

$$y(t) = X_t \phi^{(j)} + \sigma^{(j)} \epsilon_t \; if \; r_{j-1} < z_t < r_j \tag{5.14}$$

Where $X_t = (1, y_t, y_{t-1}, y_{t-2} \ldots y_{t-p})$ is a column vector of variables. In essence, the $k-1$ non trivial thresholds $(r_1, r_2, \ldots, r_k)$ divide the domain of the threshold variable z_t into k different regimes. In each different regime, the time series y_t follows a different $AR(p)$ model. When the threshold variable $z_t = y_{t-d}$ with the delay parameter d being a positive integer, the dynamics or regime of y_t is determined by its own lagged value y_{t-d} and the TAR model is called a self-exciting TAR or SETAR model.

To compare the goodness of fit of SETAR we used Smooth Transition Auto-Regressive (STAR) models. Smooth Transition Auto-Regressive (STAR) models are applied to time series data as an extension of autoregressive models; this will allow a higher degree of flexibility in model parameters through a smooth transition.

STAR model can be presented by the Eq. (5.15).

$$y_t = X_t + G(z_t, \zeta, c) X_t + \sigma^{(j)} \epsilon_t \tag{5.15}$$

Where,$X_t = (1, y_{t-1}, y_{t-2}, ..., y_{t-p})$ is a column vector of variables; $G(z_t, \zeta, c)$ is the transition function bounded between 0 and 1.

Transition function of Logistic STAR (LSTAR) model is given by Eq. (5.16).

$$G(z_t, \zeta, c) = (1 + exp(-\zeta(z_t - c)))^{-1}, \zeta > 0 \tag{5.16}$$

5.3.3 Results Analysis of Resource Forecasting with SETAR

It is confirmed from section 5.1, that there is a structural change in the data, so we decided to fit the data with SETAR model. To find the best fit we considered the following procedure:

- Define a pool of candidate models,
- Estimate the models,
- Compare the estimated models with error metrics.

We used STAR model with maximum two, three, four regimes and LSTAR model to compare with SETAR model. For convenience, we named STAR model with two, three, four regimes as STAR-2REG, STAR-3REG, and STAR-4REG respectively. Table 5.5 summarizes the comparison of STAR, LSTAR, and SETAR models using AIC, AIC_c, BIC and error metrics $RMSE$, $MAPE$ and MAE.

To compare the predictive accuracies of forecasts by SETAR model with other STAR models we used Diebold-Mariano Test (Section 4.3.5). We used SETAR model as the base model to compare with rest of the models. Table 5.6 summarizes the comparison of predictive accuracies using Diebold-Mariano Test.

Here, the significance level of the test is $\alpha = 0.05$. This is a two-tailed test with significance level of 0.05, so it must be split such that 0.025 in the upper tail and 0.025 in the lower tail. The z-value that corresponds to -0.025 is -1.96, which is the lower critical z-value and 1.96 is the upper critical z-value. The null hypothesis of zero difference will be rejected if the computed DM statistic falls outside the range of -1.96 to 1.96. From Table 5.6, we cannot reject the null hypothesis so we can conclude that all models have relatively same predictive power.

Table 5.5: Comparison of SETAR, LSTAR and STAR Models

Model	AIC	AIC_c	BIC	$RMSE$	MAE	$MAPE$
SETAR	1678.78	1678.79	1674.42	87.09	67.33	0.45
LSTAR	1679.98	1679.99	1678.22	88.09	68.33	0.46
STAR-2REG	1680.12	1681.02	1681.99	88.59	68.73	0.46
STAR-3REG	1682.72	1683.43	1684.09	89.19	69.01	0.46
STAR-4REG	1683.92	1685.87	1686.09	89.69	69.01	0.47

Table 5.6: Diebold-Mariano Test Results

Model	DM	$p-value$
LSTAR	0.3932	0.5821
STAR-2REG	0.3464	0.9236
STAR-3REG	0.7421	0.6199
STAR-4REG	0.9314	0.1656

5.4 Summary

This chapter considered the fourth objective of this research work, i.e. to investigate the presence of structural breaks and also to scrutinize the appropriateness of regime switching models for resource consumption analysis. This chapter first checked the presence of structural breaks in the data using Chow test and CUSUM test. Further, this chapter analyzed the goodness of fit of different MS-GARCH models and also analyzed the predictive accuracies of these models using DM test. Further, this chapter examined the forecasting accuracies of SETAR and STAR models. In the next chapter, we will conclude the thesis with future research directions.

Chapter 6

Conclusion and Future Directions

It is an observed phenomenon that the software exhibits increasing failure rate over time, because of a variety of reasons like resource exhaustion, the complexity of the software, nature of distribution or due to the large size of the client groups. The progressive performance degradation of the long-running software which may lead to the system crashes or undesirable hangs. The time to failure or time to exhaustion depends on the intensity of the system getting exposed to the software aging-related bugs. It may also depend on the intensity of the workload of the system and also on the total running time of the system. Resource consumption analysis is necessary for such systems because performance degradation is due to operating system's resource shrinkage.

This uncertainty in the prediction of time to failure/exhaustion is the major motivation behind this research work. To counteract the performance degradation/system failure and to increase availability, we need a proactive measure known as software rejuvenation. Most of the software rejuvenation processes are initiated when the computer performance goes beyond a certain threshold. The performance of the system, in turn, depends on the resource availability. So in order to know the time to rejuvenation the resource consumption/usage analysis is essential. Resource consumption implies the computing resources required for the execution of the computing system. These computing resources can be a percentage of CPU usage, power usage, physical memory usage, swap in/ swap out rate, network bandwidth, etc. It is observed from the literature review that memory exhaustion has contributed majorly to the system failure.

From the literature review, we found that there are different dimensions in

the research area of software performance/software aging and rejuvenation studies, namely: type of analysis, type of systems and type of resources used in resource consumption analysis. The type of analysis can be classified into three: model-based, measurement-based, and hybrid techniques. We found that the measurement based techniques have advantages over other two techniques when the scalability and accuracy are considered. We used measurement-based techniques through out this research thesis.

The next dimension is the type of system; we found that virtualized server consolidation systems are less studied when compared to other non-safety critical systems like a web server and operating system. The reason for selecting virtualized server consolidation systems is due to the increased popularity of cloud computing. Resource consumption analysis is essential in virtualized server consolidation system because the resources are used based on the client's demand hence the resource forecasting is one of the key challenging issue. It is evident from the literature review that the most popular models among the researchers studying the resource exhaustion are time series models. Some authors found that these data show nonlinear dynamic patterns like volatility and the structural changes in the data, but most of the authors did not consider the nonlinearity in the data; this motivated us to explore the analysis of volatility and the structural changes in the data. We investigated the suitability of different time series models for the resource consumption analysis of a virtualized server consolidation system, and further scrutinized the appropriateness of time-series models while dealing with data that has heteroscedasticity and structural breaks in this research thesis.

The last dimension is the type of resource used for consumption analysis. From the literature review, we observed that memory related resources usage analysis are the most popular among the software performance degradation studies, as it exhibits the shortest time to exhaustion. We used memory related data for resource consumption analysis throughout the thesis.

The first set of contributions of this thesis address the aging effect detection of virtualized server consolidation system. We established the aging effect by collecting

the resource usage data of virtualized server consolidation experimental setup and fitting the linear regression model on this resource usage data. By proving the significance of this linear fit, we established aging effect in the server virtualized system. The significance of the linear fit is established in three ways, namely: hypothesis testing, ANOVA, and by the significance of linear model parameters. From the results of the aging effect detection, we found that the average response time of the web servers significantly increased over a period when the workload remained constant for the said time. On further investigation, we found that there is free memory shrinkage during the said period. We also discovered that there is an increase in % CPU utilization and power usage during this time. From this, we concluded that the performance degradation of web servers, shrinkage of free memory accompanied by the increase in % CPU utilization and power usage is due to the aging related error propagation or simply due to aging effect. As an improvement, we may suggest analyzing the free memory of each seperate VMs.

The second set of contributions of this thesis address the appropriateness of ARIMA model for resource usage analysis. This thesis analyzed the goodness of fit for the free memory of a virtualized server consolidation system using ARIMA. We observed that ARIMA model is inappropriate for the resource forecasting in virtualized server consolidation system as the resource usage data shows patterns of nonlinearity. We analyzed the residuals of the best available ARIMA fit and found that the residuals have a non-zero mean, the residuals are not normally distributed. We observed that there is a huge change in the variance of the data. The residuals show signs of heteroscedasticity, that is the reason for the inadequateness in the fit.

The third set of contributions of this thesis address the appropriateness of nonlinear model for resource usage analysis. We analyzed the prediction errors associated with nonlinear and heteroscedastic models and carried out the residual analysis for further possible improvement in the goodness of the fit. We checked the presence of heteroscedasticity in the data and further analyzed the goodness of fit of different $ARIMA + ARCH\ /\ GARCH$ models. And then we examined the forecasting errors associated with $ARIMA + ANN$ models. We analyzed the prediction ac-

curacies of $ARIMA$ + $ARCH$ / $GARCH$ models using DM test. We compared the performance of $ARIMA$ + ANN to $ARIMA$ + $GARCH$ and $ARIMA$ using error metrics. We analyzed the error metrics and observed that ARIMA + ANN model fitted the considered data reasonably well when compared to ARIMA+ ARCH/GARCH hybrid model. But, we found that there are structural breaks in the considered data leading to inadequacy of the fit and hence these structural breaks are to be further investigated.

The fourth set of contributions of this thesis address the suitability of regime switching model for resource usage analysis. To test the structural breaks in the data, we used Chow test and CUSUM test; these tests give sufficient traces of structural breaks in the considered data set. We tried different regime switching models like MS-GARCH, SETAR, and STAR. For MS-GARCH models we used a pool of models with two, three regimes and with different distribution assumptions, like Normal, student's, and GED. We compared predictive accuracies of these models with Diebold-Mariano Test. We found that all MS-GARCH models have relatively same predictive power. By analyzing the error metrics, we could quickly determine that the predictive accuracy of the MS-GARCH models are far superior when compared to the other models considered. The next model which we considered in this work is Self-Exciting Threshold Auto-Regressive (SETAR). We considered STAR model with maximum two, three, four regimes and LSTAR model to compare with SETAR model. We compared the predictive accuracies of these models using Diebold-Mariano Test. We observed that predictive accuracies of SETAR, STAR, and LSTAR fitted the considered data set reasonably well but the predictive accuracy is inferior to that of MS-GARCH.

In summary, we started our work with establishing the software aging effect of virtualized server consolidation system. Then we analyzed the suitability of linear, nonlinear and regime switching models to forecast the resource usage data. In this process, we investigated the heteroscedasticity and structural breaks in the data set. considered data set. Finally we found that linear models are not suitable for accurately forecasting the resources in a virtualized server consolidation system. This is

due to the presence of heteroscedasticity and structural breaks in the considered data set. However, we found that heteroscedastic regime switching models are suitable for the resource forecasting in a virtualized server consolidation system.

During this research work, we found two gaps in the research area of time series analysis;

- To determine the order of ARIMA models we considered Box-Jenkins approach, but this approach more or less depends on the PACF and ACF plots for determining the order of AR and MA terms in the ARIMA model. We found that this is one of the areas to be investigated further because the ACF and PACF plots, in our case, showed no significant lags after differencing the series.

- For determining the regimes in the time series, the tests for structural break tests will not give any clue about the number of regimes present in the data set. We used a pool of regime switching models to find the best among them.

Further, we would like to try these regime switching models and study the reasons for software aging in mobile platforms like Android systems as a future research objective.

Publications

Journal Publications

1. Biju R Mohan, Ram Mohana Reddy G, Life Data Analysis of Server Virtualized System, International Journal of GEOMATE, Aug, 2017, Vol.13, Issue 36, pp.108-115

2. Biju R Mohan, Ram Mohana Reddy G, Resource Usage Prediction Based on ARIMA-ARCH Model For Virtualized Server System, International Journal of GEOMATE, May, 2017, Vol. 12, Issue 33, pp.139-146

3. Biju R Mohan, Ram Mohana Reddy G,Resource Consumption Analysis Using ARIMA-ARCH and ARIMA-GARCH Models For Virtualized Server Consolidation System,International Journal of Software Engineering and Its Applications, Vol. 11, No. 6, June 2017, pp. 1-14.

4. Biju R Mohan, Ram Mohana Reddy G, Resource Prediction in Server Virtualized System using Regime Switching Model, IET Software (Under Second Revision)

Conference Publications

1. Mohan, B.R.; Ram Mohana Reddy, G., "Software aging trend analysis of server virtualized system," Information Networking (ICOIN), 2014 International Conference on, vol., no., pp.260,263, Feb.,10-12 2014

2. Biju R Mohan, Ram Mohana Reddy G,"Analysis of Free Physical Memory in Server"Virtualized System," In the Proceedings of IEEE Sponsored 9th

International Conference on Intelligent Systems and Control (ISCO) 2015, Coimbatore, India Jan 9-10 2015

3. Biju R Mohan, Ram Mohana Reddy G, Analysis Software Aging on Power Usage, In the Proceedings of IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO) 2015, Coimbatore, India Jan 9-10 2015

4. Biju R Mohan, Ram Mohana Reddy G, Resource Prediction in Server Virtualized System, In the Proceedings of The Fifth IEEE International Conference on Communication Systems and Network Technologies (CSNT-2015), Gwalior, India April 4-6 2015 (accepted)

5. Biju R Mohan, Ram Mohana Reddy G, Life Data Analysis of Server Virtualized System, In the Proceedings of Second International Conference on Science, Engineering & Environment, Osaka City, Japan, Nov.21-23, 2016.

6. Biju R Mohan, Ram Mohana Reddy G, Resource Usage Prediction Based On ARIMA-ARCH Model For Virtualized Server System, In the Proceedings of Second International Conference on Science, Engineering & Environment, Osaka City, Japan, Nov.21-23, 2016.

References

Alonso, J., Belanche, L., and Avresky, D. R. (2011). Predicting software anomalies using machine learning techniques. In *2011 IEEE 10th International Symposium on Network Computing and Applications*, pages 163–170.

Andrade, E. C., Machida, F., Kim, D. S., and Trivedi, K. S. (2011). Modeling and analyzing server system with rejuvenation through sysml and stochastic reward nets. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 161–168.

Araujo, J., Matos, R., Maciel, P., Vieira, F., Matias, R., and Trivedi, K. S. (2011). Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds. In *2011 IEEE Third International Workshop on Software Aging and Rejuvenation*, pages 38–43.

Ariew, R. (1976). *Ockham's Razor: A Historical and Philosophical Analysis of Ockham's Principle of Parsimony*. PhD thesis, University of Illinois at Urbana-Champaign.

Avizienis, A., Laprie, J. C., Randell, B., and Landwehr, C. (2004a). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Avizienis, A., Laprie, J. C., Randell, B., and Landwehr, C. (2004b). Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33.

Avritzer, A. and Weyuker, E. J. (2004). The role of modeling in the performance

testing of e-commerce applications. *IEEE Transactions on Software Engineering*, 30(12):1072–1083.

Bao, Y., Sun, X., and Trivedi, K. S. (2005). A workload-based analysis of software aging, and rejuvenation. *IEEE Transactions on Reliability*, 54(3):541–548.

Cassidy, K. J., Gross, K. C., and Malekpour, A. (2002). Advanced pattern recognition for detection of complex software aging phenomena in online transaction processing servers. In *Proceedings International Conference on Dependable Systems and Networks*, pages 478–482.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, 28(3):591–605.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S. (2010). Software aging analysis of the linux operating system. In *Proceedings of the 2010 IEEE 21st International Symposium on Software Reliability Engineering*, ISSRE '10, pages 71–80, Washington, DC, USA. IEEE Computer Society.

Cotroneo, D., Natella, R., Pietrantuono, R., and Russo, S. (2014). A survey of software aging and rejuvenation studies. *J. Emerg. Technol. Comput. Syst.*, 10(1):8:1–8:34.

Cotroneo, D., Orlando, S., Pietrantuono, R., and Russo, S. (2013). A measurement-based ageing analysis of the jvm. *Software Testing, Verification and Reliability*, 23(3):199–239.

Durbin, J. (1960). The fitting of time-series models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 28(3):233–244.

Edson, B., Hansen, B., and Larter, P. (1996). Software reliability, availability, and maintainability engineering system (soft-rames). In *Proceedings of 1996 Annual Reliability and Maintainability Symposium*, pages 306–311.

Eto, H., Dohi, T., and Ma, J. (2008). *Simulation-Based Optimization Approach for Software Cost Model with Rejuvenation*, pages 206–218. Springer Berlin Heidelberg, Berlin, Heidelberg.

Garg, S., Puliafito, A., Telek, M., and Trivedi, K. (1998a). Analysis of preventive maintenance in transactions based software systems. *IEEE Transactions on Computers*, 47(1):96–107.

Garg, S., Puliafito, A., Telek, M., and Trivedi, K. S. (1997). On the analysis of software rejuvenation policies. In *Computer Assurance, 1997. COMPASS '97. Are We Making Progress Towards Computer Assurance? Proceedings of the 12th Annual Conference on*, pages 88–96.

Garg, S., van Moorsel, A., Vaidyanathan, K., and Trivedi, K. S. (1998b). A methodology for detection and estimation of software aging. In *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)*, pages 283–292.

Gauss-Markov. Gauss-markov theorem. `https://en.wikipedia.org/wiki/Gauss%E2%80%93Markov_theorem`. Accessed: 2017-06-26.

Ghahramani, Z. (2002). Hidden markov models. chapter An Introduction to Hidden Markov Models and Bayesian Networks, pages 9–42. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Giacomini, R. and White, H. (2006). Tests of conditional predictive ability. *Econometrica*, 74(6):1545–1578.

GLOSTEN, L. R., JAGANNATHAN, R., and RUNKLE, D. E. (1993). On the relation between the expected value and the volatility of the nominal excess return on stocks. *The Journal of Finance*, 48(5):1779–1801.

Gray, J. and Siewiorek, D. P. (1991). High-availability computer systems. *Computer*, 24(9):39–48.

Grottke, M., Li, L., Vaidyanathan, K., and Trivedi, K. S. (2006). Analysis of software aging in a web server. *IEEE Transactions on Reliability*, 55(3):411–420.

Grottke, M., Matias, R., and Trivedi, K. S. (2008). The fundamentals of software aging. In *2008 IEEE International Conference on Software Reliability Engineering Workshops (ISSRE Wksp)*, pages 1–6.

Hamilton, J. D. (1996). Specification testing in markov-switching time-series models. *Journal of Econometrics*, 70(1):127 – 157.

Hoffmann, G. A., Trivedi, K. S., and Malek, M. (2007). A best practice guide to resource forecasting for computing systems. *IEEE Transactions on Reliability*, 56(4):615–628.

Huang, Y., Kintala, C., Kolettis, N., and Fulton, N. D. (1995). Software rejuvenation: analysis, module and applications. In *Twenty-Fifth International Symposium on Fault-Tolerant Computing. Digest of Papers*, pages 381–390.

Jr., R. M., Trivedi, K. S., and Maciel, P. R. M. (2010). Using accelerated life tests to estimate time to software aging failure. In *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pages 211–219.

Koutras, V. P. and Platis, A. N. (2011). Applying partial and full rejuvenation in different degradation levels. In *2011 IEEE Third International Workshop on Software Aging and Rejuvenation*, pages 20–25.

Li, L., Vaidyanathan, K., and Trivedi, K. S. (2002). An approach for estimation of software aging in a web server. In *Proceedings International Symposium on Empirical Software Engineering*, pages 91–100.

Lyu, M. R. (2007). Software reliability engineering: A roadmap. In *2007 Future of Software Engineering*, FOSE '07, pages 153–170, Washington, DC, USA. IEEE Computer Society.

Magalhaes, J. P. and Silva, L. M. (2010). Prediction of performance anomalies in web-applications based-on software aging scenarios. In *2010 IEEE Second International Workshop on Software Aging and Rejuvenation*, pages 1–7.

Matias, R., Barbetta, P. A., Trivedi, K. S., and Filho, P. J. F. (2010). Accelerated degradation tests applied to software aging experiments. *IEEE Transactions on Reliability*, 59(1):102–114.

Matias, R. and Filho, P. J. F. (2006). An experimental study on software aging and rejuvenation in web servers. In *30th Annual International Computer Software and Applications Conference (COMPSAC'06)*, volume 1, pages 189–196.

Myint, M. T. H. and Thein, T. (2010). Availability improvement in virtualized multiple servers with software rejuvenation and virtualization. In *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement*, pages 156–162.

Nelson, D. B. and Cao, C. Q. (1992). Inequality constraints in the univariate garch model. *Journal of Business & Economic Statistics*, 10(2):229–235.

Parnas, D. L. (1994). Software aging. In *Proceedings of the 16th International Conference on Software Engineering*, ICSE '94, pages 279–287, Los Alamitos, CA, USA. IEEE Computer Society Press.

Rabiner, L. R. (1990). Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Riedmiller, M. and Braun, H. (1992). Rprop - a fast adaptive learning algorithm. Technical report, Proc. of ISCIS VII), Universitat.

Salfner, F. and Wolter, K. (2010). Analysis of service availability for time-triggered rejuvenation policies. *Journal of Systems and Software*, 83(9):1579 – 1590. Software Dependability.

Sen, P. K. (1968). Estimates of the regression coefficient based on kendall's tau. *Journal of the American Statistical Association*, 63(324):1379–1389.

Silva, L., Madeira, H., and Silva, J. G. (2006). Software aging and rejuvenation in a soap-based server. In *Fifth IEEE International Symposium on Network Computing and Applications (NCA'06)*, pages 56–65.

Silva, L. M., Alonso, J., and Torres, J. (2009). Using virtualization to improve software rejuvenation. *IEEE Transactions on Computers*, 58(11):1525–1538.

Sullivan, M. and Chillarege, R. (1991). Software defects and their impact on system availability-a study of field failures in operating systems. In *[1991] Digest of Papers. Fault-Tolerant Computing: The Twenty-First International Symposium*, pages 2–9.

Tong, H. and Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society. Series B (Methodological)*, 42(3):245–292.

Vaidyanathan, K., Harper, R. E., Hunter, S. W., and Trivedi, K. S. (2001). Analysis and implementation of software rejuvenation in cluster systems. *SIGMETRICS Perform. Eval. Rev.*, 29(1):62–71.

Vaidyanathan, K. and Trivedi, K. S. (2005). A comprehensive model for software rejuvenation. *IEEE Transactions on Dependable and Secure Computing*, 2(2):124–137.

Viterbi, A. (2006). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269.

VMware. Understanding memory resource management in vmware esx server. `https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/techpaper/perf-vsphere-memory_management.pdf`. Accessed: 2017-06-26.

Wang, D., Xie, W., and Trivedi, K. S. (2007). Performability analysis of clustered systems with rejuvenation under varying workload. *Performance Evaluation*, 64(3):247 – 265.

Weimer, W. (2006). Advanced topics in exception handling techniques. chapter Exception-Handling Bugs in Java and a Language Extension to Avoid Them, pages 22–41. Springer-Verlag, Berlin, Heidelberg.

White, H. (1980). A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4):817–838.

Yan, Y. and Guo, P. (2016). A practice guide of software aging prediction in a web server based on machine learning. *China Communications*, 13(6):225–235.

Yan, Y., Guo, P., and Liu, L. (2014). A novel hybridization of artificial neural networks and arima models for forecasting resource consumption in an iis web server. In *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pages 437–442.

Zhang, H., Wu, G., Chow, K., Yu, Z., and Xing, X. (2011). Detecting resource leaks through dynamical mining of resource usage patterns. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 265–270.

Zhao, J., Jin, Y., Trivedi, K. S., and Jr., R. M. (2011). Injecting memory leaks to accelerate software failures. In *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, pages 260–269.

Brief Bio-Data

Biju R Mohan
Research Scholar
Department of Information Technology
National Institute of Technology Surathkal
Surathkal, Mangalore
Karanataka 575025

Address of Communication

L 32 NITK Campus
National Institute of Technology Surathkal
Mangalore
Karanataka
Phone Number : 9481516999
e-mail: bijurmohan@gmail.com