CLOUD RESOURCE MANAGEMENT FRAMEWORK USING MONARCH BUTTERFLY HARMONY SEARCH AND CASE BASED REASONING

by

MOHAMED REZK AHMED MOHAMED GHETAS

**Thesis submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy**

August 2017

ACKNOWLEDGEMENTS

First and foremost, all praises to Allah for the guidance to complete this thesis. Secondly, I have to thanks my supervisor Assoc. Professor Dr. Chan Huah Yong for his guidance and support throughout this study and especially for his confidence in me. His comments and advice are very beneficial in completion of the research. I would like to thanks my wife Amira and my lovely daughters Losiana and Gana, I express my heartfelt for their support. Also I would like to express my gratitude to the School of Computer Sciences for making all this possible.

This thesis is only a beginning of my journey!

TABLE OF CONTENTS

CHAPTER 4 – IMPLEMENTATION

CHAPTER 5 – EXPERIMENTAL RESULTS AND DISCUSSION

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

QoS	Quality of Service
CP	Cloud Provider
SLA	Service Level Agreement
ROP	Resource Optimization and Provisioning
GRO	Global Resource Optimizer
RI	Resource Identifier
CPU	Central Processing Unit
VM	Virtual Machine
AWS	Amazon Web Service
EC2	Amazon Elastic Compute Cloud
PI	Proportional Integral
MBO	Monarch Butterfly Optimization
CBR	Case-based Reasoning
EM	Execution Module
PBS	Policy Based Scaling
TDS	Tier Dividing Scaling
HS	Harmony Search
MBHS	Monarch Butterfly Harmony Search
BBO	Biogeography-based optimization
SGA	Stud Genetic Algorithm

RANGKA PENGURUSAN SUMBER AWAN MENGGUNAKAN CARIAN HARMONI MONARCH RAMA-RAMA DAN PENAAKULAN BERASASKAN KES

ABSTRAK

Perkhidmatan awan telah berkembang pesat dan kebanyakannya telah menggunakan senibina pelbagai peringkat untuk fleksibiliti dan penggunaan semula. Pelbagai pendekatan berasaskan peraturan dan model telah direka untuk menguruskan kualiti perkhidmatan (QoS) untuk perkhidmatan ini. Sebahagian daripada pendekatan ini adalah bertujuan untuk meningkatkan keuntungan peruntukan perkidmatan awan. Walau bagaimanapun, ianya berdasarkan algoritma pencarian optimum di mana ia mungkin tidak berupaya untuk mencari cadangan peruntukan perkhidmatan yang terbaik dalam persekitaran awan yang berskala tinggi. Kajian ini mencadangkan rangka kerja sumber pengoptimuman dan peruntukan (ROP) yang baru untuk mengenalpasti, menyelesaikan kesesakan, dan mencapai keperluan QoS peringkat perkhidmatan bagi pelbagai perkhidmatan multi peringkat awan dan meningkatkan keuntungan peruntukan perkidmatan awan. Rangka kerja ROP terdiri daripada dua komponen utama: pengoptimum sumber global dan pengecam sumber. Kajian ini juga bertujuan untuk mempertingkatkan algoritma pengoptimumam rama-rama dan menggunakan algoritma tersebut ke dalam ROP sebagai pengoptimum sumber global. Selain itu, pengecam sumber baru telah dibangunkan menggunakan penaakulan berasaskan kes dan disertakan ke dalam rangka kerja ROP. Sebuah prototaip di platform awan telah dibangunkan, dan penjanaan beban kerja serta perkhidmatan modal pelbagai peringkat telah digunakan untuk

menunjukkan keberkesanan ROP berbanding dengan pendekatan-pendekatan lain yang telah dicadangkan dalam kajian ini. Keputusan yang diperolehi menunjukkan bahawa rangka kerja ROP boleh menjimatkan 20% daripada jumlah tuntutan mesin maya dan juga memberikan jaminan perkidmatan QoS peringkat perkhidmatan bagi perkhidmat-an yang kritikal.

CLOUD RESOURCE MANAGEMENT FRAMEWORK USING MONARCH BUTTERFLY HARMONY SEARCH AND CASE BASED REASONING

ABSTRACT

Cloud services have evolved rapidly and some have adopted a multi-tier architecture for flexibility and reusability. Various rule- and model-based approaches have designed to manage quality of service for these services. A few of existing resource management approaches aim to increase the cloud provider (CP) service provisioning profits. However, they are based on local search optimization algorithms, which may not obtain the best resource provisioning decision in a large-scale cloud environment. This research proposes a new resource optimization and provisioning (ROP) framework to detect, solve the bottlenecks, and satisfy the service-level QoS requirements of several multi-tier cloud services and to increase the CP service provisioning profits. The ROP framework consists of two main components: global resource optimizer (GRO) and resource identifier (RI). This research enhances the butterfly optimization algorithm and plugs the resulting algorithm into the ROP as a GRO. In addition, a new RI is developed using case-based reasoning and is then plugged into the ROP framework. To demonstrate the effectiveness of the proposed ROP against rule- and model-based approaches, a prototype running on a cloud platform is developed, and a workload generator and multi-tier service model are adopted. Results show that the ROP framework can save 20% of the total virtual machine demands while providing service-level QoS guarantee for critical services.

CHAPTER 1

INTRODUCTION

1.1 Background

The recent decade has marked the birth of cloud computing. Cloud computing offers a reliable, customized, and dynamic environment for hosting distributed services and applications through a conceptual layer that runs on the top of a virtual infrastructure. Organizations can avoid high up-front expenditures in a private computing infrastructure and improve the resource management and provisioning services by either outsourcing their computation to the cloud or building their own private cloud (Zafar et al., 2017). Currently, cloud providers (CPs), such as Google Compute Engine and Amazon Web Service (AWS) offer a computing platform for hosting services and applications with high availability and quality of service (QoS).

The success of the cloud computing mainly depends on how it can effectively manage the underlying resources to satisfy the time-varying performance requirements of the running services, which are characterized by QoS (e.g., throughput, response time, high availability, and security) based on the service-level agreements (SLAs) (Gullhav, Cordeau, Hvattum, & Nygreen, 2017). In this regard, an SLA is a legal contract between the CP and the service owner and states the QoS guarantees that the cloud computing platform has to provide with its running service.

Given that QoS significantly influences the growth of the cloud computing paradigm, research communities pay much attention to the QoS guarantees within the context of cloud computing. The CPs offer QoS guarantees for the hosted services and applications on the basis of the contracted SLAs to increase the service provisioning profits. However, some high-demanding services adopt a multi-tier architecture for software reusability and flexible scaling. The most significant QoS performance metric for such services is response time. Response time is a measure of the time taken to serve the request and to return the reply to the end user and is also known as end-to-end delay. Response time is measured either by the average time or 95th percentile of the time taken to serve requests over a period of time.

Providing QoS guarantees for multi-tier services is not a straightforward task because of two main reasons: one is that the workload patterns are unpredictable and continuously change over time, and the other is that the complex interaction between tiers increases the difficulty in identifying the bottlenecks and resolving them automatically (Iqbal, 2012). Therefore, the CP needs to adopt a dynamic resource provisioning and optimization approach to fulfill the obligation to the service owners regarding the SLA requirements. Given that the said services run on a shared infrastructure, the CP needs to optimize the resource provisioning between the different running services when the aggregate resource demands exceed the CP resource pool capacity to increase the CP service provisioning profits.

In summary, the CPs should adopt a dynamic resource management framework to satisfy the varying QoS requirements of the hosted multi-tier cloud services and thus reduce the SLA violation penalties and increase the service provisioning profits.

1.2 Problem Statements and Research Questions

The cloud infrastructure is shared among several multi-tier services. Some of these services have high-demanding resources and adopt a multi-tier architecture. The resource demands of these services are time varying in accordance with the concurrent users and workload patterns. The cloud infrastructure can be saturated if the aggregate on-demand resources from all services sharing the cloud infrastructure exceed the capacity of the CP resource pool. However, some services on a shared infrastructure commonly have different priority levels of resource provisioning based on the SLAs. Therefore, performance isolation and differentiation between co-hosted services need to be provided to offer QoS for critical services and increase the CP service provisioning profits.

Iqbal, Dailey, Carrera, and Janecek (2011); Lama and Zhou (2012, 2013) proposed resource provisioning approaches for multi-tier services. They employ a set of rules to identify the amount of required resources for offering QoS guarantees of the hosted multi-tier service. However, these approaches have several limitations. First, the construction of these rules incurs high computational complexity that cannot be ignored. Second, the learning speed directly influences the SLA violation rate.

Other existing resource provisioning approaches (Ashraf, Byholm, Lehtinen, & Porres, 2012; Jiang, Lu, Zhang, & Long, 2013; Urgaonkar, Shenoy, Chandra, Goyal, & Wood, 2008; X. Wang, Du, Chen, & Li, 2008) employ a performance model to describe the relationship between the target performance in terms of response time and on-demand resources. They approximate this relationship to be linear, thereby incurring a residual error during identification of on-demand resources. These approaches also result in inaccuracy if the workload deviates from the one used to identify the model parameters.

The main aim of the existing resource provisioning in multi-tier cloud service is to offer QoS guarantees. These approaches may require prior knowledge of the running services. Their performance depends mainly on adjusting the model parameters according to the changing workload pattern rate, thereby making them service dependent.

A few of existing resource provisioning approaches of multi-tier service have studied resource optimization to increase the CP service provisioning profits, but the focus is on the use of local search optimization algorithms. Therefore, these approaches may not find the best resource provisioning decision, thereby increasing the SLA violation penalties. Thus, they may be inappropriate in a large-scale cloud environment. To address these problems, this research focuses on answering the following questions:

1. How can the on-demand resource for each individual service be identified?
2. How can effective resource provisioning approach be provided when the cloud infrastructure is overloaded?
3. How effective is the proposed solution in satisfying the SLA performance level in terms of response time while increasing the CP service provisioning profits?

1.3 Research Objectives

This research aims to design a resource optimization and provisioning (ROP) framework for managing the performance of several multi-tier cloud services in relation to their SLA performance level requirements and optimizing the CP service provisioning profits. The proposed framework continuously manages the resource provisioning of running services and optimizes the resource provisioning between the hosted services. Therefore, the objectives of this research are as follows:

1. To develop a new resource identifier for reducing SLA violation in terms of response time.
2. To develop a new metaheuristic optimization algorithm for increasing CP service provisioning profits.
3. To propose a new ROP framework to simulate, integrate, and evaluate the efficiency of different resource optimizers.

1.4 Research Contributions

The proposed ROP framework for resource management differs from the existing resource provisioning frameworks such as rule- and model-based approaches. The rule-based approaches use predefined static rules (e.g., central processing unit (CPU) threshold (Iqbal, 2012)) and dynamic rules (e.g., fuzzy controller and neural network (Lama & Zhou, 2012, 2013)) to identify on-demand resources that can fulfill the SLA performance level.

The model-based approaches use network queuing theory (Urgaonkar et al., 2008) and control theory (Ashraf, Byholm, Lehtinen, & Porres, 2012) to build a theoretical performance model for identifying on-demand resources. Although these approaches can improve service performance, most of them are service dependent and may not be able to find the best resource provisioning decision for increasing the CP service provisioning profits given insufficient resources of CP.

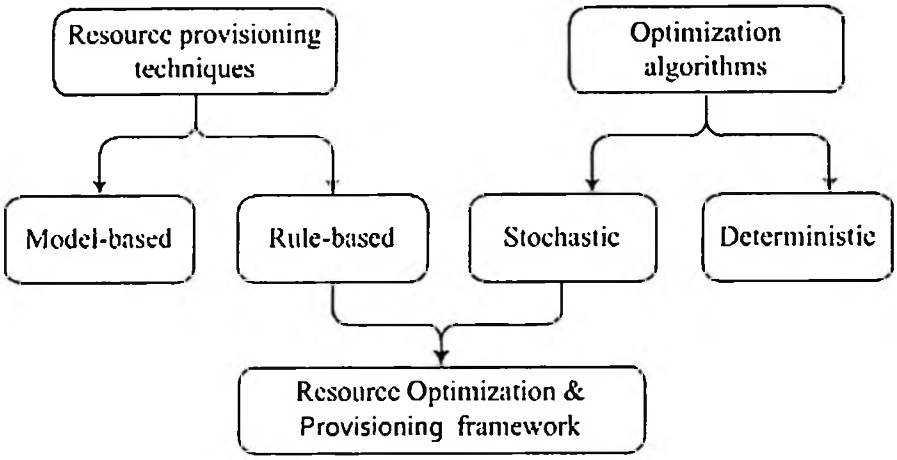

Figure 1.1: Research contribution

The existing resource provisioning approaches of multi-tier cloud services focuses mainly on providing QoS guarantees, and a few of these approaches study resource optimization when the cloud infrastructure is overloaded. However, these approaches only focus on the local search algorithms and may not be able to find the appropriate resource provisioning decision for increasing the CP service provisioning profits. Figure 1.1 illustrates the difference between the proposed framework and the existing frameworks such as rule- and model-based approaches.

The contributions of this research are as follows:

1. This research introduces a new resource provisioning technique that uses case-based reasoning (CBR) and can be plugged into the proposed ROP framework as an resource identifier (RI). The proposed resource provisioning approach provides service-level decision on the on-demand resources and the best time to provide them to satisfy the target SLA performance level.
2. This research identifies the best optimization algorithm and enhances the performance of this algorithm. The enhanced optimization algorithm can be plugged into the proposed ROP framework as a GRO to find the appropriate resource pro-

visioning decision when the aggregate resource demands exceed the CP resource pool capacity.

3. This research designs a resource management framework for multiple multi-tier cloud services running over a virtualized infrastructure. The proposed ROP framework identifies the on-demand resources for each individual service and finds the appropriate resource provisioning decision for reducing the cloud infrastructure overload. The proposed framework continuously maintains the target SLA performance level and increases the CP service provisioning profits.

1.5 Research Methodology

The design and development of the proposed ROP framework include four stages. The first stage identifies the architecture of the multi-tier service, specifies the function of each individual tier, and determines different request types that can be processed by multi-tier services. Moreover, this stage explores different techniques for providing QoS guarantees to multi-tier services and increasing the CP service provisioning profits, including admission control, service differentiation, and resource management. In addition, this stage reviews the existing resource provisioning approaches and determines the research gaps between these approaches. In accordance with the research problem and the proposed contributions, this stage explores the using of metaheuristic optimization algorithms, and case-based reasoning techniques in the cloud computing environment.

The second stage presents an overview of the proposed ROP framework. The proposed ROP framework consists of two main components: the RI and the GRO. The RI module provides service-level decisions on the on-demand virtual machines (VMs) to satisfy the target service-level QoS guarantees in terms of response time. Meanwhile, the GRO module adjusts the resource provisioning among several co-hosted multi-tier services to increase the CP service provisioning profits when the aggregate resource demands exceed the CP capacity. The ROP framework includes other helping components such as the load balancer, VM profiling agent, and monitoring agent.

A new resource provisioning method based on case-based reasoning is introduced. This method can be plugged into the ROP framework as an RI to determine the VM demands for each individual service for satisfying the target performance requirement in terms of response time. This stage also identifies the monarch butterfly optimization (MBO) algorithm as the best metaheuristic optimization algorithm. MBO is enhanced by incorporating harmony search (HS) as the mutation operator, thereby developing the so-called monarch butterfly with HS (MBHS) algorithm. The performance of the hybridized MBHS algorithm is evaluated as an optimization algorithm to solve global numerical optimization problems by use of 14 standard benchmark functions. The hybridized MBHS algorithm can be plugged into the ROP framework as a GRO to optimize the resource provisioning for increasing the CP service provisioning profits and reducing the target service-level performance violation.

The third stage develops a prototype running on a cloud platform to evaluate the performance of the proposed ROP framework and thus achieve the research objectives. The prototype includes the implementation of the RI, GRO, and other assisting components. Thereafter, different experiments are designed to evaluate the performance of the proposed ROP framework, workload generator that produces different workload patterns, RUBiS benchmark as a model for multi-tier service, and implementation and offline profiling of some existing resource provisioning approaches.

The final stage evaluates the capability of the proposed ROP framework to manage the multi-tier service performance and to increase the CP service provisioning profits through a series of experiments. The first experiment evaluates the proposed resource provisioning method as an RI of the ROP framework to satisfy the service-level performance requirement in terms of response time against other existing approaches. The second experiment evaluates the performance of MBHS as a GRO of the ROP framework to meet service-level QoS guarantee of critical services, increase the CP service provisioning profits, and reduce the target service-level performance violation penalties. Finally, the third experiment compares the performance of MBHS as a GRO with other existing resource optimization algorithms for increasing the CP service provisioning profits and reducing the service-level performance violation penalties.

1.6 Research Scope

The research scope is the resource management of multiple multi-tier cloud service running in a shared cloud infrastructure. The research focuses on the design of a resource management framework for several multi-tier cloud services to identify the on-demand resources for each individual service and to optimize resource provisioning between the hosted services. Furthermore, this research adopts read-intensive service to investigate the effectiveness of the proposed framework. This service has a two-tier architecture and runs on a homogeneous cloud infrastructure.

1.7 Research Significance

The proposed framework is a useful middleware for CPs. It enables CPs to achieve the SLA performance requirements and satisfy the service owner. The final goal is to increase the CP service provisioning profits by optimizing the resource provisioning between the hosted services. The best service provisioning profits can be achieved by identifying the best service configuration that provides high profits to the provider and satisfies the target SLA performance requirements for each individual service.

1.8 Thesis Layout

This thesis is organized as follows. Chapter 2 presents a review of existing resource provisioning techniques for multi-tier cloud service, metaheuristic optimization algorithms, and case-based reasoning method. Chapter 3 introduces the details of the proposed ROP framework and research justifications. Chapter 4 examines the implementation of the proposed ROP framework and existing resource provisioning approaches. Chapter 5 discusses and analyzes the experimental results. Finally, Chapter 6 concludes and revisits the contributions and suggests future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

This chapter describes the cloud service architecture and explores existing techniques for providing service-level QoS guarantees of multi-tier service first. Then, existing resource management approaches as well as the performance metrics to evaluate these approaches are reviewed. On the basis of the research problem and the proposed solution, the using of optimization algorithms and case-based reasoning techniques in the field of cloud computing is reviewed. Then, existing cloud platforms are reviewed. Finally, the open challenges and research gaps of the existing resource management approaches are examined.

2.2 Cloud Service Architecture

High-demanding services adopt a multi-tier architecture to offer flexibility, modularity, scalability, and reliability in deploying services (Berry, 2003). In multi-tier service, the processing of request traverses through a pipeline in which each tier receives partial processed requests from the preceding stage, performs local processing, and forwards these requests to the subsequent tier. Figure 2.1 demonstrates the architecture of an e-commerce application that consists of three tiers: web server, application, and database tiers.

The web server acts as the presentation layer and has three functions: (1) accepting/rejecting incoming requests and serving static content, (2) forwarding complex requests to the application server, and (3) receiving the response from the application server and sending a reply back to the client. Microsoft Internet Information Server and Apache are good examples of web servers (Fielding & Kaiser, 1997).

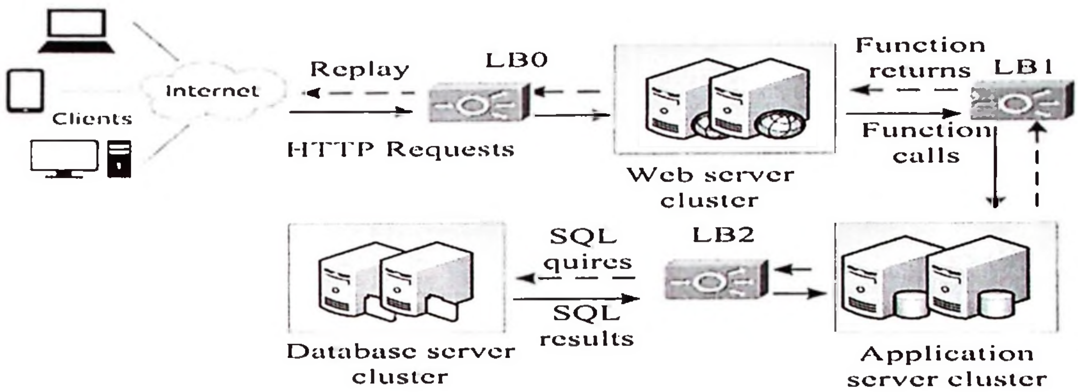

Figure 2.1: Architecture of three tier e-commerce application

The application server implements complex business logic and provides functionalities for security, session state, and database access. For example, the application server Tomcat (Tomcat, 2015) is a servlet engine container that implements Java Server Pages and Java Servlet (Community, 2015). The database tier is considered the data house, which is used to store user accounts, customer orders, and site information. Examples of database engines in a multi-tier application are Microsoft SQL, Sybase, MySQL, Oracle, and PostgreSQL.

Each service tier is ideally distributed across distinct servers. Furthermore, a tier may be clustered based on the required capacity to sustain the service performance based on the SLA requirements. For example, the web server can be run on multiple VMs, and the number of replicas should be identified to provide a sufficient capacity.

The said architecture introduces new resource management opportunities and challenges, such as the problem of identifying the bottleneck tiers and resolving them automatically, and preventing over resource provisioning by determining the on-demand resources. Furthermore, the relationship between service performance and on-demand resources is nonlinear because each tier needs different types and amounts of resources and affects the QoS in varying degrees (Kleinrock, 1975; Lloyd et al., 2013; Menascé & Almeida, 2001).

2.3 Existing Techniques for Guaranteeing QoS and Increasing CP Profits

The key resource management objective in a multi-tier cloud service is to provide QoS with least resource usage costs. The running time of the service is divided into epochs, and the length of every epoch is adjusted to balance the gain and the overhead of the QoS management techniques. Considering the fluctuation and unpredictable behavior of multi-tier service workload, providing QoS guarantee is a challenge. Therefore, a good QoS management technique not only provides QoS guarantees but also improves resource utilization. Three different techniques offer QoS guarantees: admission control, resource management, and service differentiation.

The admission control is considered the gateway of every request. It avoids the system overload caused by workload burst and provides QoS guarantees at the beginning of each epoch. The admission control identifies the number of requests that can be served under the current resource provisioning and rejects excess requests. However, the admission control design is challenging because the resources required to serve each request differ and a tradeoff between dropping and admitting requests is required.

The resource management offers QoS guarantees by scaling the running service up or down depending on the workload fluctuation. It has two types of scaling: vertical scaling and horizontal scaling. In vertical scaling, the resource management algorithm at the service layer determines the resource demands of every running VM and then the resource scheduler residing on each physical node allocates the estimated resources to the hosted VM. If the physical host has insufficient resources to resize the VM, then the resource scheduler initiates the VM migration to another physical host with sufficient resources. At present, some infrastructure providers, such as Rackspace, use vertical scaling to provide QoS guarantees for the hosted services (Rackspace, 2015)).

Horizontal scaling offers QoS guarantees by changing the number of running VMs in each tier on the basis of the workload fluctuation and the current configuration of each tier. For example, AWS and Amazon Elastic Compute Cloud (EC2) offer dozens of VM instances with different computing capabilities. In particular, vertical scaling and horizontal scaling are complementary to each other and can be applied to provide QoS guarantees.

Service differentiation offers QoS guarantees to high-priority services and improves performance of other services when the CP infrastructure is insufficient to satisfy all the on-demand resources of the running services. In other words, the service differentiation approach moves the resources between running services in case of insufficient resource to meet SLA performance requirements of the premium service class and provide improved performance for the basic service class.

2.4 Performance Metrics for Evaluation the Work in the Literature

The final objective of resource management approaches is to provide an efficient virtual server provisioning scheme with the end-to-end response time guarantee for multi-tier service. The service dependency, cost-latency tradeoff, and how the performance of multi-tier service is calculated are the three main performance metrics that are used to evaluate the existing research management approaches.

The service dependency measures how the resource management approach can adapt its parameters to handle the workload changes. In other words, the service profiling needs to be done offline for each workload pattern before the service replication and allocation. However, because multi-tier service workloads are often highly dynamic in nature, the service profiling itself can be complex and time consuming. For instance, the model based approaches are service dependent because every time the multi-tier service workload change, the values of the service model parameters have to be re-estimated.

The resource management approach has to consider not only achieving the performance requirements but also the cost of the running virtual server. Moreover, the researchers envision that the performance metric of 95th percentile response time compared to the average response time, has the benefit that is both easy to reason about and to capture the user's perception of multi-tier service performance (Lama & Zhou, 2012).

2.5 State of the Art for Guaranteeing QoS of Multi-tier Service

Numerous methods for providing QoS guarantees in multi-tier service and increasing the CP service provisioning profits have been developed. In general, these methods can be categorized into rule- and model-based approaches. The rule-based approaches use a set of rules to describe the relationship between the target performance in terms of response time and on-demand resources. These approaches are based on action selection techniques, including fuzzy control logic, reinforcement, and static machine learning. The rule-based approaches can identify on-demand resources through learning the service behavior from historical data but may not provide QoS guarantees during the learning stages.

Table 2.1: Comparison between different resource provisioning approaches

Key	Rule-based	Model-based
Techniques	1. Fuzzy control logic 2. Reinforcement learning 3. Statistical machine learning	1. Queueing network 2. Control theory
Advantages	1. Do not require an explicit performance model 2. Approximate on demand resources based on historical data	1. Offer QoS guarantee 2. Provide an explicit performance model 3. Provide rigorous and guidelines for analysis, design and evaluate the system
Disadvantages	1. Do not provide QoS guarantee	1. Queuing-based approaches are mean oriented 2. Control theory may experience a mistake of modeling dynamic workload

The model-based approaches include control theory and queuing network theory. They provide QoS guarantees, theoretical performance model and guidelines for analysis, and design and evaluation of the computing system. For example, the theoretical control techniques have been applied to non-linear computing systems to offer QoS guarantees by providing a linear performance model for multi-tier service. However, this model may experience inaccuracy when the workload deviates from those used to identify the model parameters. Table 2.1 summarizes the advantages and disadvantages of rule- and model-based approaches (Lama & Zhou, 2013).

The next two sections discuss and analyze the existing resource management approaches based on the afore-mentioned classification (e.g., rule-based and model-based) for offering QoS guarantees, increasing the CP service provisioning profits in multi-tier services, and giving rules to design and develop of the proposed ROP framework.

2.6 Model-based Resource Management Approaches

This section discusses the existing model-based resource management approaches (e.g., control theory, and queuing network theory) for providing the service-level QoS guarantees of multi-tier services.

2.6.1 Power and Performance Management Through Look-a-head Control

Kusic, Kephart, Hanson, Kandasamy, and Jiang (2009) proposed a dynamic two layer resource management framework for multi-tier application to increase the CP revenue by reducing the energy consumption, switching costs, and SLA violations. The authors identified resource provisioning as a sequential optimization problem. They employed the limited look-a-head control (LLC) and control theory to identify the new system configurations based on the number of active servers, number of VMs, required

CPU capacity to be provisioned to each VM, and number of servers to be turned off.

The results showed that LLC can achieve power savings of up to 26% with SLA violations of not more than 1.6% of the total requests over 24 h. Their proposed model is limited by that it requires prior knowledge of the shared CPU of each VM. Moreover, the resource optimization requires more than 30 min for a small system, thereby making it inapplicable to large-scale systems.

2.6.2 Cost-aware of Multi-tier Applications

Han, Ghanem, Guo, Guo, and Osmond (2014) proposed a single layer framework to address the resource scaling problem of multi-tier service in order to minimize the resource usage costs. The scaling process detects all the bottleneck tiers first and then iteratively resolves these tiers to prevent the creation of another bottleneck. For example, if the workload change trends increase the request rate, then the scaling-up algorithm identifies the tier that has the least cost as the bottleneck and adds a new server to this tier. The scaling algorithm repeats the service scale-up process until the desired response time is achieved. On the contrary, if the workload change trends decrease the request rate, then the scaling-down algorithm iteratively removes one server each time from the tiers. Notably, removing one server does not violate the SLA.

The results showed that the proposed method can successfully identify the bottleneck and requires 2–3 min to restore the desired response time. Their proposed approach is limited by that it reduces the resource provisioning costs at the service level and ignores the resource shorten at the data center level, thereby significantly affecting the resource usage costs.

2.6.3 Resource Scaling for Web Applications

Jiang et al. (2013) investigated the problem of virtual resource scaling for web applications to achieve a cost-latency tradeoff by reducing the resource usage costs and SLA violations. The authors developed a regression model to predict the web request volume in the future time unit. They employed an autocorrelation function to identify the key features of the regression model. The authors developed a performance model using queuing network theory and Marko's chain to make a decision for application scaling based on the predicted latency time. The authors employed the regression and performance models to formulate the resource scaling as optimization problems and applied an exhaustive search method to determine the on-demand resources.

The results demonstrated that the linear regression model can identify future web requests with minimum prediction error. The comparison with PEAK, PEAK ($\times 3/4$), and CAP ($\times 2$) approaches indicated that the proposed approach can improve cost saving and reduce SLA violations. This approach is limited by that the performance model is application dependent, and the prediction padding should be adjusted for each application.

2.6.4 Feedback Control Resource Management in Multiple Web Applications

Ashraf, Byholm, Lehtinen, and Porres (2012) proposed a reactive feedback control framework called ARVUE to automatically scale and deploy multiple applications on a given infrastructure and to scale the application tier as well as the physical server and thus reduce the resource usage costs. The main idea is to deploy multiple simultaneous applications on a single VM. ARVUE does not need any prior knowledge of the running applications and only uses the lower and upper CPU thresholds. The algorithm has two main components: global and local controllers. The global controller implements the scaling operations for the application server tier, and the web applications based on the CPU thresholds. The local controller logs the server- and application-level utilizations and sends the utilization data to the global controller. The authors employed additional VMs to absorb the sudden peak load and to reduce the VM provisioning delay.

The experimental results showed that sharing VM resources among different web applications can drastically minimize the number of running VMs. ARVUE is limited by that the CPU thresholds should be adjusted depending on each application behavior, the additional VMs incur excess resource usage costs, and the algorithm ignores the VM migration costs.

2.6.5 Economical Resource Provisioning of *N*-tier Applications

Xiong et al. (2011) addressed the resource management problem in multi-tier service under resource budget and performance constraint. The resource provisioning problem is divided into two sub-problems (e.g., service level, and VM level): one is the identification of on-demand resources, and the other is the partition of resources. Queuing network theory is applied to build a performance model. The application controller employs feedback with the Lagrange function to determine the optimal amount of resources that can reduce the resource usage cost with a minimum SLA performance violation. On the container level, a second controller partitions the identified resources by the application controller among the application tiers to achieve the target response time.

The experimental results demonstrated that the performance controllers saved resources up to 20% compared with utilization and equal-share approaches. Their proposed approach is limited by that it needs to solve complex optimization problem, the application controller is centralized, and the performance model may not capture the shape of the response time distribution.

2.6.6 Agile Resource Allocation of Multi-tier Applications

Urgaonkar et al. (2008) addressed the problem of dynamic VM provisioning in multi-tier applications that have long- and short-term workload variations. The objective is to determine the amount of resources to be allocated for maintaining the desired QoS. The proposed resource management framework includes a nucleus software component that resides in every server and periodically measures the performance and resource utilization. The component then delivers these measurements to the control panel. The control panel applies a queuing-based analytical model to estimate the required capacity to be allocated to each tier for satisfying the performance metrics based on the decomposition and the end-to-end delay across various tiers. The resource management framework has two distinct modules: predictive and reactive modules. The predictive module predicts the trends of workload variation based on historical observations first and then estimates the required resources to accommodate long-term change in workload.

The reactive module corrects the errors caused by the deviation of the long-term workload or by unanticipated flash crowds. However, allocating new resources is time consuming. Therefore, resources are switched from one application to another to reduce VM deployment overhead. Consequently, two approaches are employed: fixed-rate and measurement-based ramp downs. The resources in the fixed-rate ramp down are switched from under-loaded service to another in a fixed period of time. On the contrary, the measurement-based ramp down is a conservative approach and significantly depends on decreasing the rate of the resource usage of the existing session.

The experimental results showed that the proposed resource management framework accurately identifies the bottleneck tier and precisely determines the required capacity. The comparison of the proposed framework with the black-box approach showed the superiority of the former as it accounts for the replication constraints imposed on each tier. Their proposed framework is limited by that it fails to address the consolidation of data-intensive services (e.g., database scalability) and ignores the release of unused system resources.

2.6.7 Analytical Model for Resource Provisioning

Q. Zhang, Cherkasova, Mi, and Smirni (2008) investigated the problem of resource allocation in multi-tier service to satisfy the SLA performance requirement. A theoretical framework based on the regression model and queuing network theory is introduced. The regression model estimates the on-demand CPU capacity required by each transaction. The authors developed an analytical model of queues. This model uses the results of the regression model to evaluate the required resources under changing workload conditions. The mean-value analysis applied to detect the mean system throughput and end-to-end delay.

The results showed that the regression model can identify the cost of the transaction of the front-end tier with higher accuracy than the cost of the transaction of database tier. Moreover, the results from the analytical model exactly match those from the experiment. This approach is limited by that it is service dependent and cannot capture the dynamic change in workload, the predication model is overestimated over 15%, the detection of bottleneck tier is missing, and the method for adding resources for absorbing the workload spike is not provided.

2.6.8 Queuing-model-based Resource Provisioning of Multi-tier services

Liu, Heo, Sha, and Zhu (2008) proposed adaptive admission control framework based on queuing model predictor and an online adaptive control loop to identify the application capacity for achieving the target response time. The queuing model predicts the probability of accepting incoming requests that can achieve the desired response time given the current observed workload. The feedback control adjusts the admitting probability and corrects the residual error as a result of the inaccuracy of the queuing model and the change in workload. This controller adopts the regression model for identifying the relationship between the residual error and the admitting probability.

The authors evaluated the proposed approach through a test-bed experiment using the TPC-W (TPC-W, 2015) benchmark. The results showed that the proposed approach is superior to the queuing model, adaptive control (Karlsson, Karamanolis, & Zhu, 2004; Y. Lu, Abdelzaher, & Tao, 2003), and proportional–integral (PI) control (Kamra, Misra, & Nahum, 2004; Sha, Liu, Lu, & Abdelzaher, 2002) in terms of aggregating error and dropping request rate. Their proposed approach is limited by that it is based on an offline estimation of the service time.

2.6.9 Self-tuning Controller for Multi-tier Web Applications

Kamra et al. (2004) proposed a control-theoretical method for admission control in multi-tier service to keep the response time within the target value. The proposed self-tuning controller called Yaksha is composed of a Proportional Integral (PI) controller and an M/G/I processor-sharing queue. The processor-sharing queue estimates the average response time as a function of service cost and incoming request rate and adjusts the parameters of the feedback-based PI controller. The feedback-based PI controller employs a closed-loop function to estimate the number of requests to be dropped for achieving the target response time and high throughput during system overload.

The results showed that Yaksha can successfully achieve the target response time and provide high throughput under changing workload conditions. Yaksha is limited by that the adaptive controller is based on control theory, the computing applications are nonlinear, and the use of the average response time as a performance metric.

2.6.10 Virtualization-based Autonomic Resource Management

X. Wang et al. (2008) addressed the problem of resource provisioning in multi-tier services to meet different service quality targets at a minimum resource usage cost. They proposed a resource management framework that consists of self-configuration, self-optimizing, self-protecting, and self-healing modules. The self-optimizing module