Reactive Prediction Models for Cloud Resource Estimation

by

Qi Hu

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

The undersigned recommend to the

Faculty of Graduate and Postdoctoral Affairs

acceptance of the thesis

Reactive Prediction Model for Cloud Resource Estimation

Submitted by **Qi Hu**,

in partial fulfillment of the requirements for the degree of

Master of Applied Science in Electrical and Computer Engineering

_____

Chair, Yvan Labiche

Department of Systems and Computer Engineering

_____

Thesis Supervisor, Prof. Marc St-Hilaire

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Carleton University

April, 2016

Abstract

This paper provides a new reactive resource estimation model to achieve more reasonable resource allocation and higher server utilization for cloud providers. The model is flexible enough to adapt to different situations given the current server utilization, customer loyalty and price of the service. Four mathematical models are first proposed to deal with different situations. Then, a reactive model combining these four models is introduced. Simulations based on CloudSim are designed and implemented in Java. All models meet the expectations derived from the mathematical analysis in the simulation. Based on real-time simulations, the resource utilization of the combined model is improved by 18% compared to previous model.

To my loves, Xin, Cailan and Bin

Acknowledgement

I would first like to express my sincere appreciation to the help and support of my thesis supervisor, Professor Marc St-Hilaire. Your supervision and encouragement were very important for me to complete this thesis. I am very proud that I have this chance to work with you.

I would also like to thank Dr. Mohammad Aazam for his useful suggestions and comments on my work. You are always patient and nice to guide me and inspire me. It is also a great experience to work with you.

I would like to express my gratitude to my parents; Cailan and Bin. You are always supporting me no matter what happens.

Last but not least, I want to thank all my friends and colleagues who encourage me and motivate me all the time.

Table of Contents

List of Tables

List of Figures

List of Acronyms

AOP	Average Overall relinquish Probabilities
API	Application Programming Interface
CSC	Cloud Service Customer
CSM	Cloud Service Model
CSP	Cloud Service Provider
FCFS	First-Come First-Served
FPRRM	Flat Period Reservation-Reducing Method
GFS	Google File System
HDFS	Hadoop Distributed File Systems
HMM	Hidden Markov Model
IaaS	Infrastructure as a Service
ICAF	Inter-cloud Architecture Framework
ICCMP	Inter-cloud Control and Management Plane
ICFF	Inter-cloud Federation Framework
ICOF	Inter-cloud Operation Framework
ILS	Integrated Local Search
InP	Infrastructure Provider
LTPM	Linear Trend Predicting Method
MQMPM	M/M/1 Queuing Model Prediction Method

PaaS	Platform as a Service
PM	Physical Machine
SaaS	Software as a Service
SLA	Service Level Agreement
SOP	Service Oriented relinquish Probabilities
VM	Virtual Machine

Chapter 1

Introduction

In this chapter, we first introduce the topic of cloud computing and describe the problem statement. Then, the main research objectives are outlined followed by the research contributions. Finally, a general overview of the thesis is presented.

1.1 Problem Statement

Nowadays, cloud computing has been more and more important in the IT industry. Individuals and small companies utilize computing resources, network resources, and storage resources from the cloud instead of buying servers to run their own businesses. However, with the development of technology, the number of services that is available to customers keeps growing to the point where a single cloud provider may not be able to provide everything everyone needs. Thus, one cloud may have to request another cloud or even multiple clouds for additional services. This expanded system, called inter-cloud computing or cloud federation will allow communication and shared resources among clouds. This involves adding a cloud broker, whose main function is to negotiate between the involved parties and find the best service for each request with specified service level agreements and allocate resources to the users. The broker also bills a small percentage for its own intermediary services.

In a cloud computing environment, power consumption is a big issue as there are more and more large-scale data centers for the increasing demands of resources. Large-scale data centers consume enormous amounts of electrical power even when resource utilization is low. According to recent research, the average resource utilization is lower than 50% [1]. In some areas, the average resource utilization is even lower. According to research, customers usually overestimate the amount of the required resources, which leads to wasting a large amount of the requested resources [2]. All the problems mentioned above are related to the area of cloud computing in general.

In cloud computing, customers can submit two types of requests: reserved services and on-demand services. This thesis will focus on the second type of request. On-demand requests can also be divided into two subtypes. One could use the cloud to reserve virtual machines with specific requirements in order to accomplish specific tasks. In this case, the provider will allocate what was requested and will charge accordingly. More recently, people have been using the cloud to request different services. For example, when a customer submits a request to stream a video from Netflix, different video quality (i.e. SD, HD, UHD, etc.) could be allocated depending on the resource estimation algorithm.

Customers may overestimate the requirements (duration) for their request and relinquish the service before the service expires. In this case, providers have to recycle the resources and try to reallocate them, which costs money and time. More importantly, these resources may remain unutilized and will consume electric power. Therefore, it is important to find a resource management mechanism to help solve this problem. A good resource allocation scheme could help cloud providers to increase server utilization and reduce unnecessary power consumption.

In a cloud computing environment, a customer may ask for some services and have some requirements about the quality of those services. When the request is submitted, it will first be analyzed by the broker (or provider) and the amount of allocated resources will be decided. If the customer asks for a specific amount of resources, then the provider has to allocate the same amount of resources to the customer.

In [3], Aazam et al. propose a broker-based resource estimation model in which the broker will do resource estimation based on records of customers. The model is innovative in the sense that it treats customers differently according to their behaviors. However, if two customers have the same behavior, they will always be treated the same way, regardless of the evolving cloud environment. It would be ideal to have resource estimation models assign resources not only based on customer behavior but also based on the cloud provider environment.

1.2 Research Objectives

The main objective of this thesis is to come up with an improved resource estimation model for cloud broker in order to improve server utilization. The model should also be flexible enough to adapt to different situations that can be seen in a cloud provider environment. More precisely, we want to:

- Propose and analyze (mathematically) different strategies to assign resources under various criteria such as server utilization, the amount of allocated resources, and the customer experience.
- Simulate a cloud computing environment using the CloudSim toolkit and implement the different strategies mentioned above.
- Compare the mathematical results with the simulation results of the different strategies.
- Propose a new reactive prediction model that will use different strategies to assign resources. The model should be able to adapt to various conditions such as current cloud utilization and the loyalty of customers in order to make accurate assignments of resources.
- Compare the performance of the proposed model with the performance of the resource estimation model in [3].

1.3 Contributions

The main contributions of this thesis are as follows:

- Build a reactive prediction model with several modules for resource estimation. The broker will choose different resource estimation strategies according to several factors: customer loyalty, current server utilization and the requested services. For each scenario, the model will utilize different modules to adapt to the situation and make resource estimation reasonable for each kind of customer.

- Compare the different modules with each other and the old resource estimation model. Each module will be illustrated and analyzed

mathematically. The overall allocated resources, the overall server utilization and customer experience will be included in the analysis. They will also be used for comparison between the new model and the old model.

- Design simulations based on the CloudSim toolkit to determine the performance of the new model in different situations. The CloudSim toolkit will be extended for simulation. Each module, including the old model, will be simulated in some specific scenarios. The outputs will be collected for comparison and analyzed to see if the simulation results meet the mathematical analysis. Besides, real-time simulations will be built with one cloud service provider and requests with random durations from different customers with random loyalties at random time stamps. The simulation will run with both the old resource estimation model and the new model. After the simulation, the trend of server utilization for these two models will be illustrated.

1.4 Thesis Overview

The rest of this thesis is organized as follow. Chapter 2 provides some background information and an overview of the state of the art research that is relevant to the thesis. Chapter 3 briefly introduces the model presented in [3] and proposes new models to do resource estimation. It will be seen that each model can be mapped to a specific situation and that the combination of all these models refer to what we call the “reactive model”. Chapter 4 describes the simulation environment and provides an analysis of the simulation results. Finally, Chapter 5 summarizes the thesis and outlines future work on the topic.

Chapter 2

Background and Related Work

This chapter will discuss important background information and key techniques about cloud computing plus some popular cloud products such as Amazon EC2 and Google App Engine. After that, a literature review of recent published research about resource management in cloud computing and inter-cloud computing will be presented.

2.1 Background

2.1.1 Cloud Computing Architecture

Generally speaking, the architecture of cloud computing consists of four layers: the hardware layer, the infrastructure layer, the platform layer and the application layer [4]. Table 2.1 lists those layers and their corresponding types of services with some examples of products.

- **The hardware layer**: The hardware layer manages the physical resources of the cloud and is implemented in data centers. It contains a lot of hardware equipment such as servers, hard disks, etc.
- **The infrastructure layer**: The infrastructure layer has a pool of storage and computing resources by partitioning the physical resources using virtualization technologies such as Xen [5], KVM [6] and VMware [7]. This layer is essential for cloud computing because many key features are available due to virtualization technologies.
- **The platform layer**: The platform layer is on the top of the infrastructure layer. This layer is built to minimize the burden of the virtual machine (VM) containers where users deploy applications.
- **The application layer**: The application layer is at the highest level of the architecture and includes the cloud applications. These cloud applications can achieve better performance, availability and lower operating cost [8].

Table 2.1: Cloud Computing Architecture

Layers	Type of Service	Examples
Application Layer Business Applications, Web Services, Multimedia	**Software as a Service (SaaS)**	Google Apps, Facebook, YouTube, Saleforce.com
Platforms Layer Software Framework, Storage (DB/File)	**Platform as a Service (PaaS)**	Microsoft Azure, Google App Engine, Amazon S3
Infrastructure Layer Computation (VM), Storage (block)	**Infrastructure as a Service (IaaS)**	Amazon EC2, GoGrid, Flexiscale
Hardware Layer CPU, Memory, Disk, Bandwidth		Data Centers

As for cloud-based services, there are three types of service models: i) software as a service, ii) platform as a service and iii) infrastructure as a service.

Software as a Service (SaaS) means customers do not need to install the software in their machine. They can simply run the software in the cloud using the web browser and the cloud provider is responsible to install and maintain the software.

Platform as a Service (PaaS) is a service which provides users with application platforms and databases. For software developers, they don't need to install different operating systems (e.g. Linux, IOS) to program for different tasks once they have this service.

For **infrastructure as a Service (IaaS)**, the cloud provider acquires the physical computing resources underlying the service, including the servers, networks, storage and hosting infrastructure. The cloud provider runs the cloud software necessary to make computing resources available to the IaaS cloud consumer through a set of

service interfaces and computing resource abstractions, such as virtual machines and virtual network interfaces [9]. Amazon EC2 is a good example of IaaS [10].

2.1.2 Distributed File System

Distributed file systems are designed and implemented to meet the rapidly increasing demands of data processing needs. There are several distributed file systems for large distributed applications such as Google File System (GFS) [11] and Hadoop Distributed File System (HDFS) [12] which can meet the storage needs nowadays and provide efficient, reliable access to data using large clusters of commodity servers.

The Google File System is built from many components which are not expensive and often fail so it is also able to detect, tolerate and recover from failures. It stores a small number of large files but also supports small files. The workloads include large streaming reads and small random reads and it also allows the clients to do concurrent appending and use high sustained bandwidth.

As Figure 2.1 shows, GFS has one master server and multiple chunkservers. Files are divided into chunks which are fixed-size and each chunk is stored into chunkservers on local disks. Each chunk has a size of 64 MB and is replicated on several chunkservers for reliability. The master server executes all name space operations, manages chunk replicas in the system and also has a function called garbage collection. When a file is deleted, the master logs it and the file is just renamed to a hidden name which includes the deletion timestamp. It is only removed during the master's regular scan of the file system namespace. In a heartbeat, the chunkserver communicates with the master server and then deletes the replicas of such chunks [11].

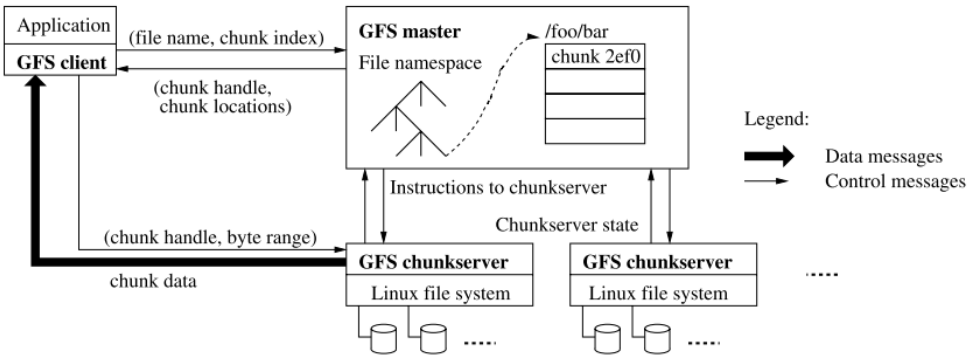

Figure 2.1: Architecture of the google file system [11]

Hadoop Distributed File System is highly fault-tolerant and is deployed on low-cost hardware. It is suitable for applications which have large data sets and enable streaming access to file system data [12]. Similar to the Google File System, it stores data on diverse nodes which use a block protocol specific to HDFS to serve blocks of data over the network. It provides access to all content from a web browser or other types of clients and data nodes can communicate to each other to help the management of data [4].

2.1.3 Parallel Programming

Users are able to access more computing resources through cloud computing techniques without knowing low-level details. However, the heterogeneity and the frequent changing of cloud environments make it difficult to manage and operate; the input data is usually large and the computations have to be distributed across thousands of machines to complete in a reasonable time limit. The issues of how to parallelize the computation, distribute the data, and handle failures become problems. So parallel programming is necessary for cloud computing to reduce the completion time of each task because programmers can identify work units and dispatch them to different processors [13].

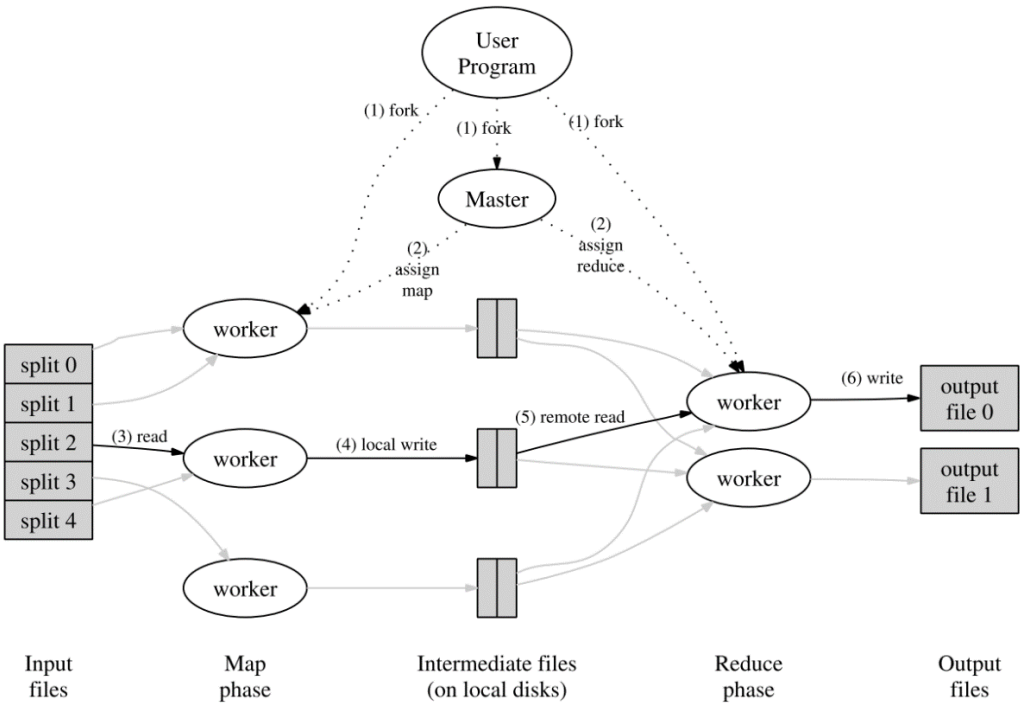

Figure 2.2: Execution overview [14]

MapReduce [14] is a software framework introduced by Google to enable automatic parallelization and distribution of large-scale computations. The MapReduce process takes a set of input <key, value> pairs, and produces a set of <key, value> pairs. As shown in Figure 2.2, the process includes two functions: Map and Reduce. The Map function groups together all the intermediate values associated with the same intermediate key and gives them to the Reduce function, which merges together these values to form a smaller set of values.

2.1.4 Inter-Cloud Computing

Current cloud computing providers have several data centers at different geographical locations over the world to better serve customers at different locations. However, a data center does not have infinite resources and it will not be able to serve new requests from customers when it saturates. In other words, it might happen that one data center is over-utilized while another data center has many idle resources. Furthermore, cloud service providers are unable to predict geographic distribution of cloud customers and therefore, the distribution of services needs to adjust in response to any changes in the cloud computing environment [15]. Consequently, clouds

should interconnect in order to provide scalability, efficiency and flexibility by sharing the services and resources with other clouds. In inter-cloud computing, the cloud service providers will dynamically resize their provisioning capability according to the change of the workload by leasing resources from other cloud providers. Every cloud operates as a part of the cloud federation [16].

2.1.5 Cloud Broker

As discussed in the last section, a single cloud cannot always fulfill the requests or provide the required services. There comes inter-cloud computing, a situation when two or more clouds have to communicate with each other. In inter-cloud, there is an intermediary known as an inter-cloud broker (or simply broker) which is an entity responsible to introduce the cloud service customer (CSC) to the cloud service provider (CSP) and vice versa. Through the interface of the cloud broker, multiple clouds can be managed and share resources. The main purpose of the broker is to help the customer find the best provider and service according to his needs and the service level agreement (SLA). The cloud broker earns its profit by satisfying requirements of both parties. The cloud broker uses some methods, such as a repository for data sharing and integration across data sharing services, to achieve the best possible deal and service level agreement between the provider and the customer [17]. The broker typically makes profit by taking reward from the completed deal or by the spread. The spread is the difference between the price at which the broker buys from the cloud provider and the price at which the customer pays to the broker [3]. Figure 2.3 shows the communication of CSC, CSP and the broker.

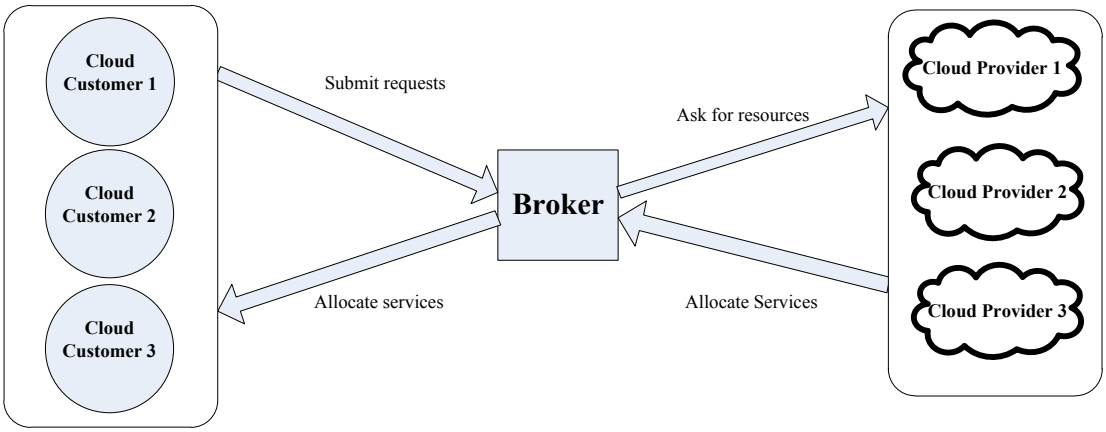

Figure 2.3: Communication between CSP, broker and CSC

A cloud broker is consisted of APIs and a standard abstract API which manages cloud resources from different cloud providers. Different modules perform a specific task in the broker's architecture, as Figure 2.4 shows. The major components of the broker are introduced as follows:

- **Service registration manager**: The service registration manager is responsible to register a service when the broker receives a service from a cloud provider.
- **Discovery manager**: The discovery manager is responsible to discover suitable resources for customers according to their needs.
- **Monitoring manager**: The monitoring manager monitors the service provisioning process including when to start and end services, etc.
- **Deployment manager**: The deployment manager deploys any software or hardware infrastructure for cloud providers such as any new module, new device, etc.
- **Match-maker**: The match-maker matches cloud providers with customer's request according to different matching strategies or algorithms.
- **Profit calculation**: Profit calculation calculates the profit for cloud providers.
- **Refund manager**: The refund manager manages issues about reimbursement. When a customer needs reimbursement, the refund manager will calculate the amount of reimbursement and reimburse the customer.
- **Customer manager**: The customer manager manages the type of devices, the

operation system, etc.

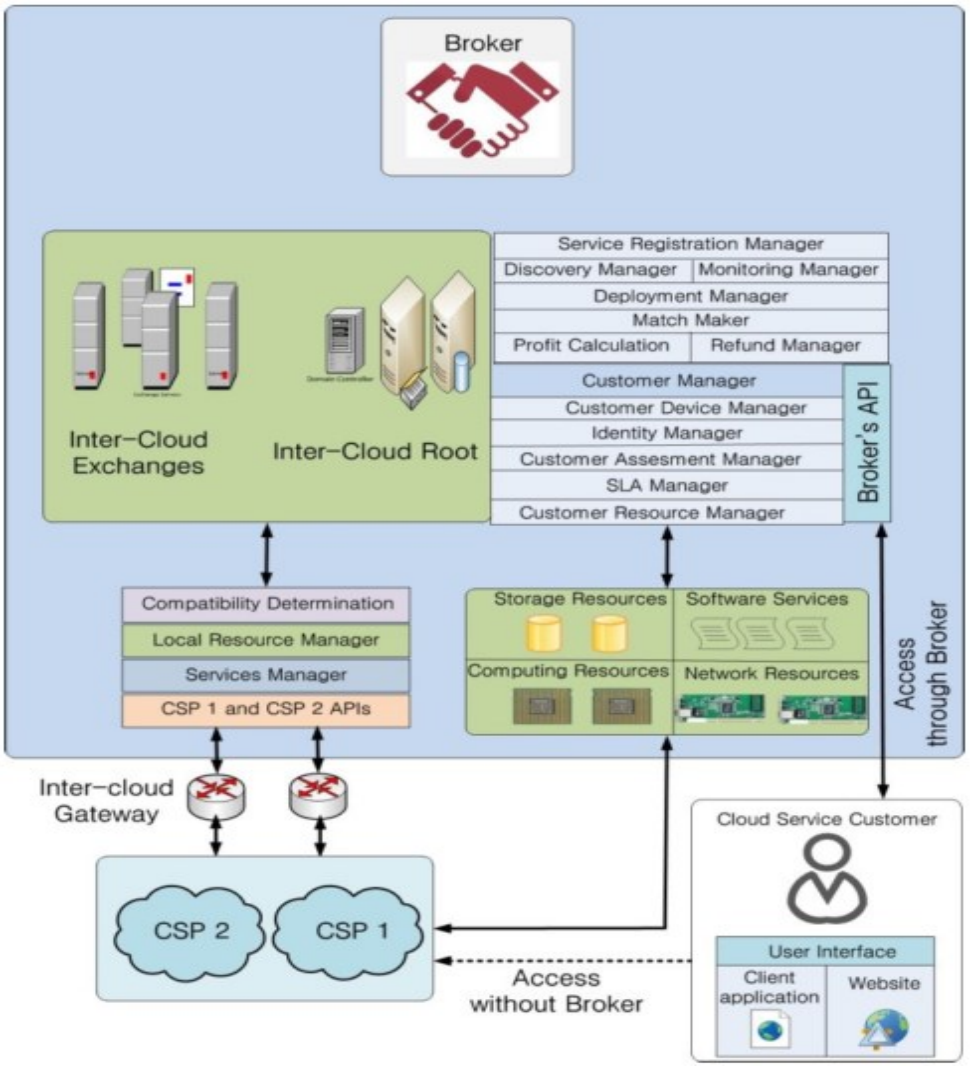

Figure 2.4: The architecture of the cloud broker and the process of communication [3]

- **Identity manager**: The identity manager handles the customer verification and admission control.
- **Customer assessment manager**: The customer assessment manager assesses the loyalty, previously utilized services, profit earned and other information of the customer.
- **SLA manager**: The SLA manager manages the provision of SLA, including SLA negotiation, SLA violation, and so on.
- **Customer resource manager**: The customer resource manager manages

resources allocated to a particular customer.

- **Compatibility determination**: Compatibility determination checks if the request of a customer is compatible with the available infrastructure.
- **Local resource manager**: The local resource manager manages resources required for local processing for the broker, which means the broker may be a service provider on its own.
- **Services manager**: The service manager manages local services utilized by the broker.

Cloud providers can interact with the broker through the inter-cloud gateway and the API. The inter-cloud gateway is a service frontend for cloud providers to interact with the broker and provide management and monitoring of the APIs. For cloud customers, they can directly access cloud providers without broker or with broker using the broker's API [18].

The cloud broker can also be used in other fields such as mobile cloud computing environment. For example, the authors in [19] and [20] propose an optimized task offloading model with a centralized broker. They optimize the task scheduling model based on user defined constraints. The centralized broker implements the model and achieves the reduction of energy consumption of mobile devices.

2.1.6 Resource Management in Cloud Computing

Cloud computing is designed to achieve a wide range of resource sharing from cloud service providers to cloud customers. In cloud computing, resource utilization and power consumption are two important factors which are highly coupled. Specifically, resources with a low utilization rate will still consume a huge amount of energy [21]. A recent study shows that the average resource utilization is lower than 50% [1]. In this case, an effective strategy of resource management is needed to increase the efficiency of resources and avoid the unnecessary waste of power consumption. Thus, cloud computing resource management plays an important role in the cloud computing environment. In cloud computing systems, resource management can be

divided into 1) resource discovery, 2) resource scheduling, 3) resource matching and 4) resource monitoring. Resource discovery is the process of searching available resources for a requested service. Resource scheduling is defined as the process of arranging and scheduling the available resources for all the requests. Resource matching is to match the corresponding resources, while resource monitoring is responsible for monitoring the task and resource state process before task completion [22]. Moreover, resource management includes different processes in different situations. For example, in inter-cloud computing environment, resource allocation is included which is used to allocate resources to different cloud customers according to their needs. Good resource management strategies could hide the heterogeneity of cloud computing resources, improve the server utilization and avoid unnecessary waste of resources and power consumption.

2.1.7 Commercial Products

Some commercial products of cloud computing are introduced in this section. Customers can use those resources to implement their software and they do not need to maintain the servers. Those products make money by leasing resources to customers and maintain the resources for customers.

- **Google App Engine**

Google App Engine [23] is a platform as a service offering which allows users to build and run applications on Google's infrastructure. With App Engine, there are no servers for users to maintain. It supports apps written in Java, Python, PHP and Go. It allows customers to run large-scale applications and all the applications are run in a relatively independent environment with security so that the Google App Engine is able to distribute requests according to different traffic demands. The Google App Engine also provides persistent storage, automatic scaling and load balancing to customers [23].

- **Amazon EC2**

Amazon Elastic Compute Cloud [24] is a web service providing computation, storage and other functionalities. It allows organizations and individuals to rent virtual

computers to deploy their own applications and services on an on-demand basis. Users can completely control their computing resources and quickly change capacity as their requirements change. A user can boot an Amazon machine image to create a virtual machine, which Amazon calls an “instance”. Customers are able to place instances in multiple locations [4] [24].

2.2 Related Work

This section describes a representative set of existing related research in the area of cloud computing. More precisely, we will focus on inter-cloud computing and resource management in cloud computing.

2.2.1 Inter-Cloud Computing

The increasing demand for cloud computing services has resulted in more heterogeneous infrastructure, making interoperability an area of concern. Due to this, it becomes a challenge for cloud customers to select the appropriate cloud service provider and hence it ties them to a particular CSP. This is where inter-cloud computing comes into play. The purpose of inter-cloud computing is to allow smooth interoperability between clouds, regardless of their underlying infrastructure. Individual service customers can seek their own service providers. This allows users to migrate their workloads across clouds easily. Cloud brokerage is a promising aspect of inter-cloud computing.

Regularly, applications can use standard Internet protocols and platforms for their interaction and management. However, when they are operated on a distributed multi-provider cloud-based infrastructure, they need a new architecture for inter-cloud. In [25], Demchenko et al. develop the inter-cloud architecture framework (ICAF) that addresses problems of integration and interoperability in multi-domain heterogeneous cloud-based applications and facilitates interoperable and manageable cloud federation. The authors list the goals of the proposed ICAF:

- ICAF should support communication between cloud applications and services in different service layers, cloud domains and heterogeneous platforms.

- ICAF should create a common inter-cloud control plane for better cloud services and network integration.
- ICAF should provide a framework for heterogeneous inter-cloud federation.
- ICAF should facilitate interoperable and measurable intra-provider infrastructures.
- ICAF should support the business models of existing cloud providers.

Based on the above requirements, the authors define ICAF including four components that are responsible for different issues:

- Multilayer Cloud Service Model (CSM) combines common cloud service models such as IaaS and PaaS in one multi-layer model with inter-layer interfaces.
- Inter-cloud Control and Management Plane (ICCMP) supports the interaction of cloud-based applications and services.
- Inter-cloud Federation Framework (ICFF) describes infrastructure components for independent cloud domains federation.
- Inter-cloud Operation Framework (ICOF) includes functionalities to support multi-provider infrastructure operation.

The proposed framework provides a basis for research in inter-cloud architectures. It refers to some existing and widely accepted solutions and splits the functionalities of the framework.

In small areas, a centralized architecture of inter-cloud computing environment might be desired, where clouds establish connectivity and collaborate among themselves. While centralized approaches are efficient, the authors in [26] argue that the centralized architecture will not work efficiently when the system is extended to a large-scale system. Sotiriadis et al. study the advantage of a decentralized versus centralized meta-broker architectures and the result of the meta-broker solution shows the performance level of average execution time for different customers who submit their requests concurrently. However, this problem is studied under the First-Come First-Served (FCFS) service policy and the authors do not mention other scenarios in the paper.

In cloud federation, it is important to know when and how should a cloud trades

resources with other clouds to maximize the net profit. Li et al. design an algorithm based on double auction mechanism for virtual machines trading among the clouds [27]. The double auction mechanism is individual rational and ex-post budget balanced. Moreover, they design a dynamic resource trading and scheduling algorithm which is responsible to find out the real value of virtual machines during the auction, turn on and off servers according to the current electricity prices and schedule job arrivals with different service level agreements.

Leivadeas et al. [28] focus on networked cloud computing environments with multiple infrastructure providers (InPs). They propose a novel request partitioning approach based on integrated local search (ILS) which facilitates the cost-efficient and online partitioning of user requests among eligible cloud service providers. The partitioning phase is followed by the final mapping phase, which follows the basic principles of the methodology in [29]. Moreover, they provide a thorough evaluation of the proposed overall framework on a simulated networked cloud environment and compare against an exact request partitioning solution. However, the paper does not discuss the visibility of the broker on the substrate network resources, which is critical for the efficiency of virtual network embedding across multiple substrate providers [30].

Wang et al. develop a smart cloud brokerage service that serves cloud user demands with a large pool of computing instances that are either dynamically reserved or launched on demand from IaaS clouds [31]. They propose two algorithms separately for long-term prediction and short-term prediction; and also dynamic strategies for the broker to make instance reservations for minimizing its service cost. Based on the model, cloud users can minimize their costs by choosing among different pricing options based on their own demands. The approach discussed in [31] does not consider advance reservation while provisioning resources. In addition, this system does not consider broker profit as a factor in the modeling part.

The broker is important in inter-cloud computing. In this thesis, it is also used as an important part of the architecture. Moreover, the proposed models for the resource estimation will be processed within the broker.

2.2.2 Resource Management in Cloud Computing

Nowadays, the number of large-scale data centers is growing significantly and the complexity of the network infrastructure is also increasing. More importantly, the power consumption of those data centers is enormous. In 2010, Google data centers used about 2.26 million megawatt hours of electricity and generated 1.46 million metric tons of carbon dioxide [32]. Additionally, building a data center leads to excessive establishment expenses as data centers are usually built to serve infrequent peak loads resulting in low average utilization of the resources. As mentioned previously, in cloud computing, resource utilization and power consumption are highly coupled. The data center will consume a huge amount of energy even when resource utilization is low. Recent research shows that the resource utilization is lower than 50% on average [1]. In addition, cloud customers have a fluctuating behavior and they have different loyalty for the service provider. In other words, some disloyal customers may relinquish resources before the scheduled end time. Some solutions need to be developed to improve resource management in order to reduce unnecessary cost and improve the resource utilization while satisfying the service level agreement.

Performing analysis on large-scale trace logs is fundamental to deriving realistic models. In [2], Moreno et al. propose an approach to derive realistic workload models. They first analyze a 30 day trace log from the Google Cloud to derive the statistical parameters of the proposed model. An exhaustive analysis of the data has been performed at three different levels: coarse-grain, cluster and intra-cluster. They have found observations and conclusions as follows:

- Modeling user behavior is a key factor when characterizing cloud workloads. Their analysis shows that user behavior has a great influence on workload characteristics (e.g., resource utilization and number of tasks) and consequently the cloud environment.
- Workloads are various in different observation periods. Their analysis has revealed that task and user dimensions differ significantly on a daily basis.
- The cloud environment does not show obvious periodical behavior. The analyzed

data reveals that there is no strong correlation between the amount of work and specific periods of time. This indicates the dynamicity that exists in cloud environments and the diversity of users and their strong influence on the workload.

- Users usually overestimate the amount of the required resources. The intra-cluster analysis shows that in over 90% of the cases, users tend to overestimate the amount of the required resources, wasting in some cases near to 98% of the requested resources.

Based on this analysis, the authors develop a novel method for characterizing workloads that considers cloud workload in the context of both user and task in order to derive a model to capture resource estimation and utilization patterns. The derived model assists in understanding the relationship between users and tasks within workload, and enables further work such as resource optimization, energy-efficiency improvements, and failure correlation.

Beloglazov et al. present a decentralized architecture of the energy aware resource management system for cloud data centers [33]. They propose three stages of continuous optimization of virtual machine placement: reallocation of multiple system resources according to current resource utilization, optimization of virtual network topologies between virtual machines and virtual machine reallocation according to thermal state of resources. Then, they present simulation-driven evaluation of heuristics for dynamic reallocation of virtual machines using live migration in the first stage. In the simulation part, several policies are used for simulation like single threshold policy and two-threshold policy. However, the authors do not mention any details about the parameters of each policy, like the value of the threshold and the reason for using that value. In addition, for the virtual machine reallocation problem, there is no explanation of the second part which is determining new placement of virtual machines.

Gao proposes an improved ant colony algorithm to solve resource allocation issues [34]. He adds energy consumption into the basic ACO model and sets the limit of the pheromone in each cycle. However, he does simulation based on the

environment of local domain. Amit et al. propose a scheduling algorithm for deadline sensitive leases based on Haizea [35]. They introduce swapping and backfilling first, and then proposed an algorithm applying swapping and backfilling for rescheduling the leases to increase acceptance of leases. The results show that by applying swapping and multiple slots concepts, the number of accepted leases increases compared to the existing algorithm. However, they do not implement backfilling in their simulation.

Shi et al. use a M/M/1 Queuing Model Prediction Method (MQMPM) to simulate the web service modeling, and propose a resource prediction algorithm to dynamically provision resources with better performance [36]. They first use the Linear Trend Predicting Method (LTPM) to predict the resources, and adopt a Flat Period Reservation-Reducing Method (FPRRM) to adjust the number of reserved resources to avoid waste of resources. According to their experiment results, the combination method can better reduce the delay during the fast increasing period and can reduce unnecessary high resource reservation in flat and smooth period. However, the performance of the prediction is not good when the sequence increases rapidly.

In [37], Lee et al. propose a two-phase approach to solve the reservation problem. The first phase considers long-term resource reservation based on a mathematic model. Specifically, mathematical equations are derived for calculating an upper bound of the optimal number of reserved long-term resource in terms of minimizing the expected long-term operation cost. The second phase is the short term dynamic allocation using Hidden Markov Model (HMM) to predict resource demand and allocate VM resource adaptively based on the prediction. Delay of resource provisioning is also considered such that the impact of resource provision delay could be reduced. The authors consider the allocation delay in their approach. However, they do not explain clearly their simulation process and some of the parameters in the simulation part are not mentioned in the model.

Wang et al. develop an energy conserving resource allocation scheme with Prediction for cloud computing systems [38]. This scheme can predict the trend of coming tasks and their features; and the system can react by shutting down a physical

machine (PM) or starting up a new PM according to the trend.

Dabbagh et al. propose an integrated resource management framework that reduces energy consumption in cloud data centers [39]. They first use *k*-Means to cluster virtual machine requests into *k* categories. Then the framework uses stochastic Wiener filter prediction to estimate the workload of each cluster and reduces energy consumption by switching unneeded physical machines to sleep mode. The framework is based on real Google traces collected over a 29-day period from a Google cluster containing over 12,500 physical machines. However, they just use one trunk of the traces as the training data set and one trunk as validation data set. For privacy reasons, some parameters are not provided by Google, which may influence the accuracy of the prediction. In addition, the authors add a safety margin for situations of under-estimation and over-estimation. However, the prediction does not perform very well.

Majumdar et al. describe broker-based system on a new framework that performs proactive auto-scaling. The proposed broker-based system determines the number of allocated resources by predicting the amount of requested resources in the future [40]. It also supports both on-demand and advance reservation requests. The authors utilize the machine learning algorithm to predict workload in the future with past workload as the training dataset. Besides, a new price model is proposed to increase profit for an intermediate broker and cost saving for cloud customers. However, the characteristics of the requests are hard to predict since they are from different types of customers with different behaviors (as discussed earlier and mentioned in [41]). The relinquishment of services and reimbursement are not considered in this system. In addition, they do not give explanations on some of the workload parameters for simulation and the machine learning algorithms for prediction.

In [3], Aazam et al. propose a broker-based resource estimation model to help customers find the most suitable provider and the service according to customer's need. It is the first time that resource estimation is based on the behavior of the customer. In this paper, the resource estimation model will predict users' relinquish probability according to their histories. Thus, the broker will calculate resources to be

allocated to users based on the prediction model and decide the price of the allocated service. Besides, the pricing mechanism of the model is based on giving incentive to more reliable customers so that there will be more loyal customers in the transactions as time goes by. Based on the resource estimation model, they also add refunding system with considerations of quality degradation of services in [42]. This thesis uses the same cloud computing architecture in [3] and proposes a new resource estimation model with several modules for different scenarios based on the model in [3]. The analysis of [3] and the new model will be introduced in detail in next chapter.

Chapter 3

The Reactive Prediction Model (RP Model) for Cloud Resource Estimation

The new resource estimation model presented in this chapter is based on the model described in [3]. To better understand the model in this paper, we will first introduce it and analyze it in details. Then, new models will be proposed and compared to each other mathematically. As will be seen, each model will have its pros and cons and they will fit into different circumstances. As a result, a combination of the previous models is proposed and we refer to this model as the reactive prediction model. Finally, different parameters and functions, such as the pledge system, are proposed to make the new model closer to real-life situations and more accurate.

3.1 The Resource Estimation Model as Described in [3]

Instead of assigning resources equally to all users, the model in [3] is using the concept of relinquish probability in order to assign resources. In other words, loyal users will be assigned more resources than non-loyal users. The authors propose two models: 1) a resource estimation model to help the broker decide how much resources should be allocated to the user and 2) a pricing model to help the broker to calculate the price of the allocated resource based on the history of the customer.

3.1.1 Resource Estimation Model

Cloud service customers contact the cloud broker to get a given service at the best price. The cloud broker negotiates with the cloud service providers and provides resources to the customer using the resource estimation model. In addition, the resource estimation is based upon the history of customer behavior. This makes sure that customers with an inconsistent history do not get assigned more resources than they need in order to avoid losing too much money. The authors in [3] formulated the prediction of the required resources as shown in equation (1).

$$R = \Psi_{pi} * \left(\left(1 - \bar{x}(P_i(L|H)_s)\right) - \sigma^2\right) * (1 - \Omega_i) \quad (1)$$

$$R \in \{CPU, storage, memory\}$$

In the equation above, R represents the required resources which can be CPU, storage or memory; Ψ_{pi} is the basic price of the resource which is typically decided when the contract is negotiated. $\bar{x}(P_i(L|H)_s)$ is the average value of the Service Oriented relinquish Probabilities (SOP) of a particular customer giving up the requested resource. In this model, a customer relinquishing resources means that the customer stops using the service before the scheduled end time. For example, a customer may be assigned resources for his service for 10 hours but he stops using it after 5 hours. In this case, the relinquish probability for this request is 50%. In this model, the customers are categorized into two types, one having low (L) relinquish probability and the other having high (H) relinquish probability. Where,

$$0.1 \leq L \leq 0.5, 0.5 < H \leq 1 \quad (2)$$

If the customer is requesting the resource for the first time, it is assumed that this new customer is 'somewhat' loyal, so the default SOP will be set to 0.3 which is the average probability of low relinquish probability (0.1 to 0.5). When a customer requests more than one service, the resources are summed up for this customer. Since customers can have fluctuating behavior in using resources, the authors use σ^2 as the variance of SOP to determine the actual SOP of each customer.

$$\Omega_i = \bar{x}\left(\bar{x}\left(\sum_{k=0}^{n} P(L|H)_k\right), P(L|H)_{last}\right) \qquad k > 0 \quad (3)$$

Ω_i, the Average Overall relinquish Probabilities (AOP), is calculated based on the overall history (regardless of the service) of the customer with the cloud provider. It is different from SOP which only calculates the average relinquish probability of the customer using the same service. In this equation, k is the number of the requests the customer made and $\bar{x}$ stands for the means of the records. If previous record exists, then $k > 0$ and it will take the average of k records of relinquish probabilities. Then it takes the average again with the relinquish probability of the "last" request. The last activity is the most recent record that the customer had, and the reason that the average is taken twice is because the last record is given additional priority in this

equation so that the value of AOP will be close to the recent behavior habit of the customer. If the customer is new with no historical data (i.e. $k = 0$) then AOP will be set to 0.3 for the same reason explained for the SOP.

The cloud broker determines future resource requirements for each customer according to equations (1), (2) and (3). This model helps cloud brokers to decide correctly how much resources should be allocated to customers. This way, the model can also help with the management of power consumption. Customers and providers can save electricity and money [3]. .

3.1.2 Pricing Model

Two pricing methods for different types of customers based on their historical behavior are proposed in [3]:

$$\rho_{S_{P(L)}} = \int_0^t (\Psi_{p_i} + \mu_L + \Omega * \beta) \tag{4}$$

$$\rho_{S_{P(H)}} = \int_0^t (\Psi_{p_i} + \mu_H + \Omega * \beta) \tag{5}$$

$\rho_{S_{P(L)}}$ is the price for the requested service S by a customer who has a low relinquish probability P_L. Similarly, $\rho_{S_{P(H)}}$ is the price for customers who have high relinquish probability. β is the broker service ratio set by the cloud broker. For example, if it is 10%, then the broker will charge 10% of what the provider earns from the service. Ω is the average overall probability.

$$\mu_L = \frac{\Psi_{p_i} * P_L}{\delta} \tag{6}$$

$$\mu_H = \frac{\Psi_{p_i} * P_H}{\delta} \tag{7}$$

μ_L is the decision variable for users with relinquish probability P_L and μ_H is the decision variable for users with relinquish probability P_H. They are calculated using equations (6) and (7) where δ stands for the total of the profit earned so far by the CSP from this particular customer. If the customer is new, δ will be set as the profit to be

earned by the service currently requested, which is represented as δ_c.

3.1.3 Analysis of the model

The resource estimation model and the pricing model introduced in [3] are novel and very interesting. However, we believe the model has some flaws as pointed out below:

- In their simulation, the price of the service is used to differentiate between services. In fact, the broker will allocate a general amount of resources to the customer without specifying what kind of resources (e.g., memory, storage, etc.) is included. We believe that this is not very realistic as real cloud providers have to map the services to real physical resources. According to the model in [3], if two services have the same price, they are treated as if they were using the same amount of resources even if they are actually using two different kinds of services.
- The authors use the variance of SOP (σ^2) in Equation (1) to help determine the actual behavior of each customer in case of the confusion caused by the very fluctuating resource demand behavior of the customer. However, the variance is a value that indicates how far a set of numbers is spread out so it cannot make the result more accurate using the average number plus or minus the variance even if the value of the variance is very small.
- The resource estimation model uses SOP and AOP to decide how much resources should be allocated, where SOP and AOP have the same weight. In fact, SOP should have more weight because it represents the average of the service oriented relinquish probabilities of a particular customer of giving up the same resource that is currently being requested [3].
- The model will assign resources based on the relinquish probability of the customer. This means that if two customers with the same relinquish probabilities ask for the same service at different point in time, they will always get the same amount of resources. Although this model is better than a flat assignment policy (everyone gets the same amount of resources), we believe it could be made even more dynamic.

- Some important parameters are not considered when assigning resources to users. For example, the current cloud utilization and the type of service could have an important impact on the amount of resources that is allocated.

Finally, the paper only partially presents the simulation parameters and this makes it difficult to regenerate similar results. Also, only partial results are presented which does not show the complete picture of how the model behaves.

3.2 New Reactive Prediction Models

Given the limitations previously mentioned, this section makes modifications to the model proposed in [3]. Then, new individual models are proposed and analyzed in order to assign resources. Finally, these individual models are integrated into a bigger resource assignment model referred to as the reactive prediction model described below.

In the old model, the authors do not mention the type of resources required for each request in the simulation. In the new model, the customer's request is modified to be more practical. *N* will be used to stand for the overall request from a customer and it will include the amount of requested CPU (N_{CPU}), the amount of requested Memory (N_{Memory}) and the amount of requested storage $(N_{Storage})$. When a customer submits a request to the broker, the broker will analyze the request and obtain the necessary amount of resources required for delivering the service. A request from a customer is split into three parts at the broker's side, the amount of CPU, memory and storage. These types of resources stand for three fundamental characteristics customers pay for with cloud services: compute, data transfer out and storage [43]. They are common resources deployed in several cloud providers such as Amazon Web Services, Dropbox and Microsoft Azure. Since the request is more specific, the whole model will be more useful, allowing more efficient use of cloud services.

As previously mentioned, the price of the service and the variance of SOP in Equation (1) are not reasonable because the price of the service cannot perfectly describe a request from a customer and the variance of SOP cannot make the resource

estimation more accurate. As a result, the equation after removing Ψ_{pi} and σ^2 and adding *N* is simplified as shown below:

$$R = \sum_{i=0}^{n} \begin{cases} N * (1 - SOP) * (1 - AOP) \\ 0 \end{cases} \tag{8}$$

$$R \in \{CPU, storage, memory\},\ N \in \{CPU, storage, memory\}$$

In Equation (8), *N* is the requested resources and *R* represents the allocated resources. *SOP* is the average of the relinquish probabilities of this particular service, *AOP* is the average of the relinquish probabilities of all services from the same cloud provider. The way to calculate *SOP* and *AOP* is basically the same but it needs some small changes. For example, suppose customer1 with *SOP* = 0.1, *AOP* = 0.9 and customer2 with *SOP* = 0.9, *AOP* = 0.1 ask for the same service. According to Equation (8), they will be allocated the same amount of resources. This unreasonable situation happens because *SOP* and *AOP* have the same weight. As mentioned previously, *SOP* should outweigh *AOP* because *SOP* is more related to the current request. Considering this point, the new way to calculate the overall relinquish probability is described by the equation below. As can be seen, the weight of SOP is two times the weight of AOP which makes sure that *SOP* will have a more important impact when deciding the amount of resources to allocate.

$$x = \frac{2}{3} * SOP + \frac{1}{3} * AOP \qquad 0 \leq x \leq 1 \tag{9}$$

In Equation (9), *x* stands for the overall relinquish probability which is composed of *SOP* and *AOP*. The range of the relinquish probability is between 0 and 1 (instead of 0.1 to 1 as in the old model) to make it include all possibilities so that the model will be more comprehensive. When *x* = 0, it means that the customer has never relinquished resources in the past. In addition, the way to calculate *SOP* and *AOP* is the same as in the model discussed in [3]. It is worth noting that the ratio could be changed, but the goal is to put more emphasis on the value of *SOP*.

Based on the modified formulation from above, the next subsections will introduce and illustrate various models in order to assign resources. Then, the models will be compared with respect to different criteria such as the overall amount of allocated resources, the overall resource utilization and the customer's experience.

3.2.1 Model 1

The first new modified model is given below.

$$R = N * f_{(x)} \tag{10}$$

$$f_{(x)} = (1 - x)^2 \tag{11}$$

$$R \in \{CPU, storage, memory\},\ N \in \{CPU, storage, memory\}$$

In equations (10) and (11), *R*, *N* and *x* are introduced in equations (8) and (9). When the broker receives a request from a customer and the customer's record from the provider, it can calculate the amount of resources to be allocated. The most important part of the new model is $f_{(x)}$. Obviously, the larger the value of $f_{(x)}$ is, the more resources will be allocated.

Figure 3.1 is a plot of Equation (11). Clearly, this model allocates most resources to loyal users while disloyal customers get less resources, which helps increase the utilization. When a customer's loyalty increases, which means *x* decreases, the value of $f_{(x)}$ increases faster. This gives a good incentive for customers to become more loyal.

On the other hand, $f_{(x)}$ is generally low which means that customers will get much less resources than what they requested for their services. For example, if a customer with $x = 0.3$ submits a request such as Equation (11), $f_{(x)} = 0.49$, then the customer will get less than 50% of what was asked. In this case, most customers will not be satisfied even if they are loyal. Besides, as the customer's loyalty decreases, as seen in Figure 3.1, $f_{(x)}$ drops very fast at first and then drops slowly when *x* increases. When *x* increases from 0.3 to 0.4, $f_{(x)}$ drops from 0.49 to 0.36. However, when *x* increases from 0.7 to 0.8, $f_{(x)}$ drops from 0.09 to 0.04. This means that loyal customers will lose more resources than disloyal customers when *x* increases and at the same time, disloyal customers will not get enough warning.

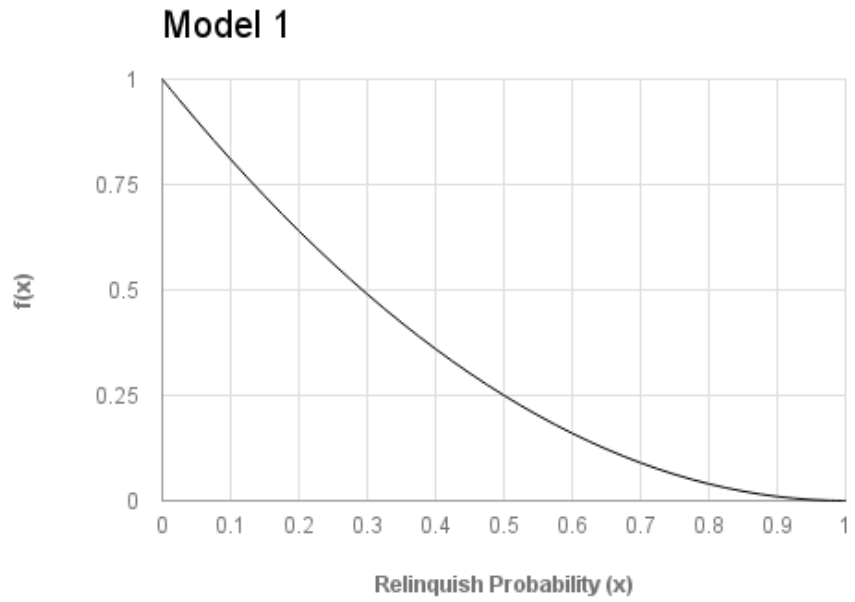

Figure 3.1: The relation between relinquish probability and *f(x)* of model 1

Overall, this model may be suitable for situations where resources are scarce since this model will allocate most resources to very loyal customers while disloyal customers will get much less. But for other circumstances, many resources will still remain idle, which is a waste of resource and power.

3.2.2 Model 2

$f_{(x)}$ of model 2 is described as shown below:

$$f_{(x)} = 1 - x^2 \tag{12}$$

In the equation above, *x* has the same meaning and range as *x* in model 1. From Figure 3.2, we can see that this model allocates much more resources than model 1. Besides, as customers become increasingly disloyal, they will begin to lose much more resources than loyal users. That would be a good warning for disloyal customers. For customers who become more loyal, the allocated resources increase significantly at first and then less so. This is a good incentive for disloyal customers to become more loyal. For loyal users, the overall amount of allocated resources is large enough to satisfy most customers' needs, so it will not influence customer satisfaction even if the amount does not increase fast.

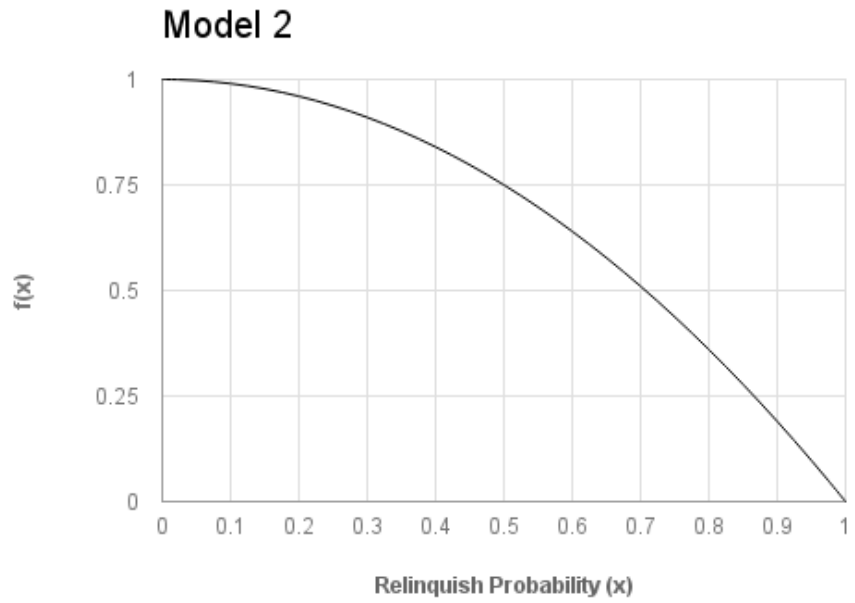

Figure 3.2: The relation between relinquish probability and *f(x)* of model 2

In general, this new model allocates much more resources for customers with different loyalties. This means that the model will be suitable when the cloud provider is underutilized. However, for scenarios where resources are tight, this new model may not work well since it allocates a significant amount of resources to customers who are not very loyal ($x > 0.5$).

In short, model 1 and model 2 are suitable for different situations. In the next two sections, the previous two models will be combined in different ways to satisfy other scenarios.

3.2.3 Model 3

For model 3, $f_{(x)}$ is a piecewise function as presented in Equation (13) and Figure 3.3.

$$f_{(x)} = \begin{cases} 0.5 + 0.5 * (1 - 2x)^2 & 0 \leq x \leq 0.5 \\ 0.5 - 2(x - 0.5)^2 & 0.5 < x \leq 1 \end{cases} \quad (13)$$

As can be seen from the figure, model 3 is a combination of model 1 and model 2. The first function is the shrunk version of model 1 and the second one is the shrunk version of model 2. The model combines the characteristics of both models and therefore eliminates some of the weaknesses.

From Figure 3.3, we can see that the overall amount of allocated resources is less than model 2 but larger than model 1. Besides, from the junction of two functions (0.5, 0.5), when x decreases, $f_{(x)}$ is increasing faster and faster. Similarly, when x increases, $f_{(x)}$ is dropping more and more quickly. Overall, it combines some advantages of the first two models and it will be more suitable for ordinary scenarios where resources are not excessive but adequate. However, for customers who have x around 0.5, there is not much difference when x changes. For instance, when x changes from 0.3 to 0.7, $f_{(x)}$ drops from 0.58 to 0.42. The incentive (or warning) is obviously not enough for customers in the middle when they are becoming more loyal or disloyal. But the good thing is that for some new customers who may have fluctuating behaviors for a period of time, this model could let them have relatively stable outcomes and the providers would not lose much money since when x is around 0.5, $f_{(x)}$ is not high and does not change much. This model may be good for situations where customers are new or services are expensive and long but requested by unreliable customers.

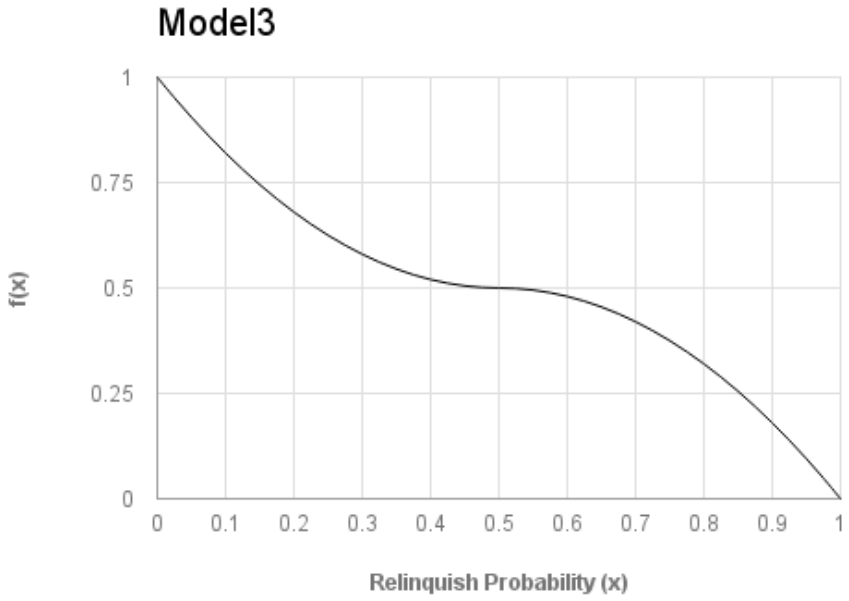

Figure 3.3: The relation between relinquish probability and *f(x)* of model 3

3.2.4 Model 4

Similar to model 3, model 4 combines the strategies from model 1 and model 2 but in

a different way. $f_{(x)}$ can be expressed as shown in Equation (14):

$$f_{(x)} = \begin{cases} 1 - 2x^2 & 0 \le x \le 0.5 \\ 0.5 * (2 - 2x)^2 & 0.5 < x \le 1 \end{cases} \quad (14)$$

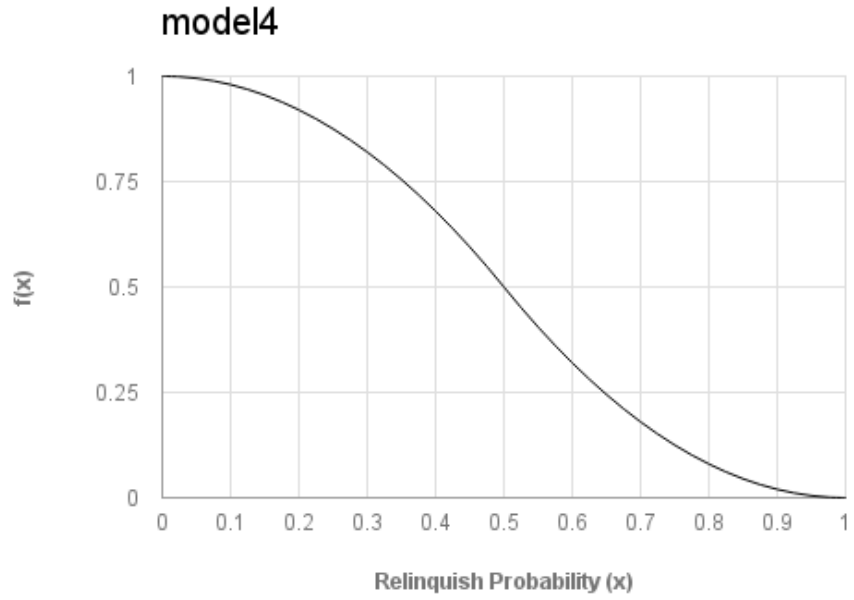

Figure 3.4: The relation between relinquish probability and *f(x)* of model 4

As Figure 3.4 indicates, the overall amount of allocated resources is about the same as model 3 and the amount of resources allocated to loyal customers is far more than what disloyal customers get. The difference is at the junction (0.5, 0.5), $f_{(x)}$ rises or drops fast at first but slowly as *x* goes down or up. This means that the behaviors of customers in the middle (i.e. *x* is near 0.5) will have a huge impact on the amount of resources allocated to them. This model is a good model for situations where the server utilization is moderate. In this model, loyal users get most of the resources while users with bad history get much less. For other customers who are in the middle level, the model is also attractive because they will have big gains if they have better behaviors.

3.2.5 Control Model

The last model is a control model introduced to compare with the other four models mentioned before. It is a flat model, which means the broker will allocate whatever

the customer asks for. We can represent it as model 5:

$$f_{(x)} = 1 \tag{15}$$

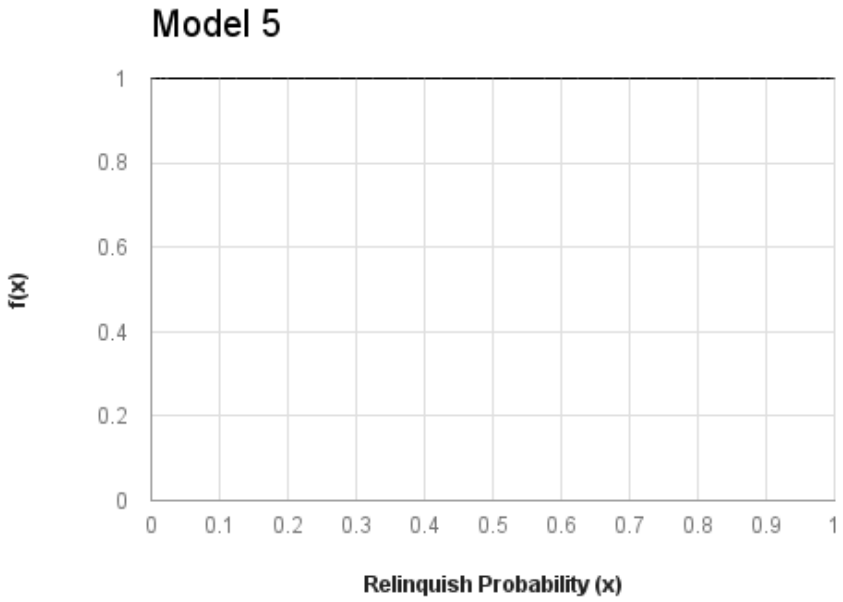

Figure 3.5: The relation between relinquish probability and *f(x)* of model 5

In model 5, as Figure 3.5 indicates, customers get whatever they want no matter what their histories are. This may cause a huge waste of resources and money in real situation.

In the next section, the models are compared in a mathematical way using different metrics such as the overall amount of allocated resources, the overall resource utilization and the customer's experience, etc. According to these metrics, the pros and cons of these models will also be analyzed.

3.2.6 Comparison of the models

In this section, the requested resource (*N)* is set to 1 so that for any customer, the requested resources will be the same according to Equation (10). The duration of the request is also set to 1, which is the unit time. Then, integration is used to calculate the overall allocated resources and resource utilization. It is assumed that under each model, customers with relinquish probability evenly distributed from 0 to 1 will ask for the same amount of resources and the number of each kind of customers is equal.

The configuration of those parameters can help us find out the comprehensive performance of each model. Those metrics are calculated by the following equations:

$$AR = 1 * \int_0^1 f_{(x)} dx \tag{16}$$

$$RR = 1 * \int_0^1 (f_{(x)} * x) dx \tag{17}$$

$$UR = AR - RR \tag{18}$$

$$U = \frac{UR}{AR} = \frac{AR - RR}{AR} = \frac{\int_0^1 \left(f_{(x)} * (1 - x)\right)}{\int_0^1 f_{(x)}} \tag{19}$$

In equations (16) to (19), *AR* stands for the overall amount of allocated resources, while *x* stands for relinquish probability and we assume that each customer will relinquish with the same probability as his overall average relinquish probability.. $f_{(x)}$ represents the percentage of allocated resources for customers with relinquish probability *x*. In this case, the summation of $f_{(x)}$ for each customer will provide the overall amount of allocated resources (*AR*), which equals to the area under the curve of each model. When a customer relinquishes the request, it means he will relinquish the whole service before the scheduled end time rather than relinquish part of the resources. Therefore, $f_{(x)} * x$ represents the relinquished service and the integration of $f_{(x)} * x$ for each customer will be the overall amount of relinquished resources (*RR*). *UR* can be understood as the actually utilized resources among the allocated resources which equals to AR minus *RR*. *U* stands for "service utilization" and represents how long the service is actually used compared to the scheduled duration. Table 3.1 summarizes the values of those parameters for each model.

For each model, the amount of resources of the allocated service and the utilized service are also shown in Figure 3.6.

From Table 3.1 and Figure 3.6, we can see that model 1 allocates the minimum amount of resources while model 2 allocates services with the most resources of all (except model 5 which is a control model). The *AR*s in model 3 and model 4 are similar and they are between the *AR*s of model 1 and model 2. Model 5 gives whatever the customer asks for. As for utilization, even though model 1 allocates less

than the other models, it has the highest resource utilization. Model 2 has the same utilization as model 3 and the utilization is near model 4's, while model 5 has the minimal utilization as expected. As for the customer's experience, as mentioned before, the previous four models suit for different circumstances and types of customers. For the control model, since the loss is obviously much larger than other models, it will not be used in the later simulation.

Table 3.1: Comparison of Different Models

Model	Allocated Service	Relinquished Service	Utilized Service	Service Utilization
Model 1	$\frac{1}{3}$	$\frac{1}{12}$	$\frac{1}{4}$	$\frac{3}{4}$
Model 2	$\frac{2}{3}$	$\frac{1}{4}$	$\frac{5}{12}$	$\frac{5}{8}$
Model 3	$\frac{1}{2}$	$\frac{3}{16}$	$\frac{5}{16}$	$\frac{5}{8}$
Model 4	$\frac{1}{2}$	$\frac{7}{48}$	$\frac{17}{48}$	$\frac{17}{24}$
Model 5	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

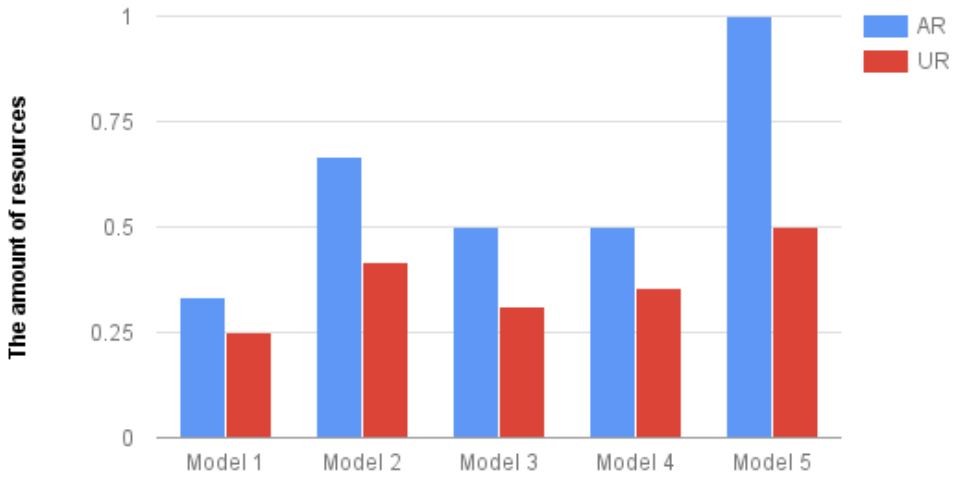

Figure 3.6: The allocated service and the actually utilized service of each model

For the customer's experience, it can also be estimated by comparing the amounts of resources allocated to different types of customers. A customer is regarded as a loyal customer if his overall relinquish probability is lower or equal to 0.5; while a disloyal user's overall relinquish probability is larger than 0.5 (see Equation (2)) . Sometimes, when the server is over-utilized or there are not enough resources, it is preferred to allocate more resources to loyal customers since it will increase the resource utilization. We also calculate the amount of resources allocated to these two kinds of customers as shown in Table 3.2.

As Table 3.2 indicates, loyal customers in model 1 will get most of the resources, followed by model 4. These models could be suitable for over-utilized situations since the resource utilization will increase since more resources are allocated to loyal customers. Model 5 is the fairest model since the loyal users and disloyal users get the same amount of resources. Loyal customers will receive two times the amount of resources compared to what disloyal customers receive in models 2 and 3.

Table 3.2: Resources Allocated to Different Types of Customers

Model	**Resources of Loyal Users**	**Resources of Disloyal Users**	**Ratio**
Model 1	$\frac{7}{24}$	$\frac{1}{24}$	7:1
Model 2	$\frac{11}{24}$	$\frac{5}{24}$	11:5
Model 3	$\frac{1}{3}$	$\frac{1}{6}$	2:1
Model 4	$\frac{5}{12}$	$\frac{1}{12}$	5:1
Model 5	$\frac{1}{2}$	$\frac{1}{2}$	1:1

In this section, several models were proposed and analyzed. These mathematical models will make the resource estimation more realistic and concrete. The customer's request is more specific and some parameters in the resource estimation model are

removed or modified to make resource estimation more accurate. Each new model has its own pros and cons and it was shown in what situations each model performs best.

3.3 Combined Model

In [3], the data center broker adopts only one model regardless of the situation of the cloud computing environment. It reduces the burden of the broker, but in real life, one model is definitely not enough to deal with so many complicated scenarios. Therefore, a combined model comprised of the 4 models introduced earlier (i.e. model 1 to model 4) is proposed to improve the performance of the broker. The combined model is called “reactive” because it can react to the environment of the cloud provider by adjusting the amount of allocated resource. In other words, in the reactive model, the amount of allocated resources depends on the current environment of the cloud and on the behavior of the customers. . This way, the resource estimation will be more effective and reduce unnecessary loss.

In this section, according to some metrics such as the server utilization and the history record of each customer, some situations will be classified as shown in Figure 3.7 and they will be analyzed and matched to different modules according to their features and requirements to make sure the provider can make reasonable profit and the customer can receive the service he deserves.

When a customer submits a request to the broker, the latter will classify the request and then perform a match making process between both the customer and the cloud after analyzing the environment variables. Below are the steps that the broker will have to do before deciding on how much resources should be allocated. These steps are also shown in the decision tree shown in Figure 3.7.

(1) Check the server utilization data. The server utilization includes information on the processing load on servers relative to the maximum server capacity. Here a provider is regarded as under-utilized when the server utilization is lower or equal to 25%, while an over-utilized provider has server utilization larger or equal to 75% and the provider in between is called a moderately-utilized provider.

(2) For moderately-utilized servers, verify if the customer is a new customer or an

existing customer. If the customer is a first time customer, the broker will automatically use the default value as his average relinquish probability (i.e. SOP and AOP).

(3) For existing customers, verify if the customer is loyal or disloyal (as mentioned before, the customer is regarded as a loyal user if his overall average relinquish probability is lower or equal to 0.5 while a disloyal customer has an overall average relinquish probability higher than 0.5). It is worth noting that even a loyal user may relinquish most of his allocated resources for some instances because the overall average relinquish probability cannot reflect the customer's behavior for each single instance as many uncertainties prevail in real life.

(4) If the customer is disloyal, then check the details of the requested service to decide if the service is expensive or not. The standard refers to the Amazon pricing system. If the price of the requested service is higher or equal to 0.462 dollar, the service will be regarded as an expensive service.

After performing the above analysis, the broker can end up with six different outcomes which are labeled situation 1 to situation 6 in Figure 3.7. The requirements and the suitable models will be analyzed for each situation.

Situation 1: This situation happens when a customer is matched to an under-utilized server. Since the server is under-utilized, it has many idle resources. In this case, resource utilization is not the first concern of the cloud provider. For example, a supermarket will reduce the price of food which is about to expire to attract more consumers since the market can at least earn some money instead of throwing it away. Similarly, it is better to allocate as many resources as possible instead of keeping those resources unutilized, even if some users are not loyal. According to this, model 2 will be the most suitable model for this situation because it generally allocates more resources than other models.

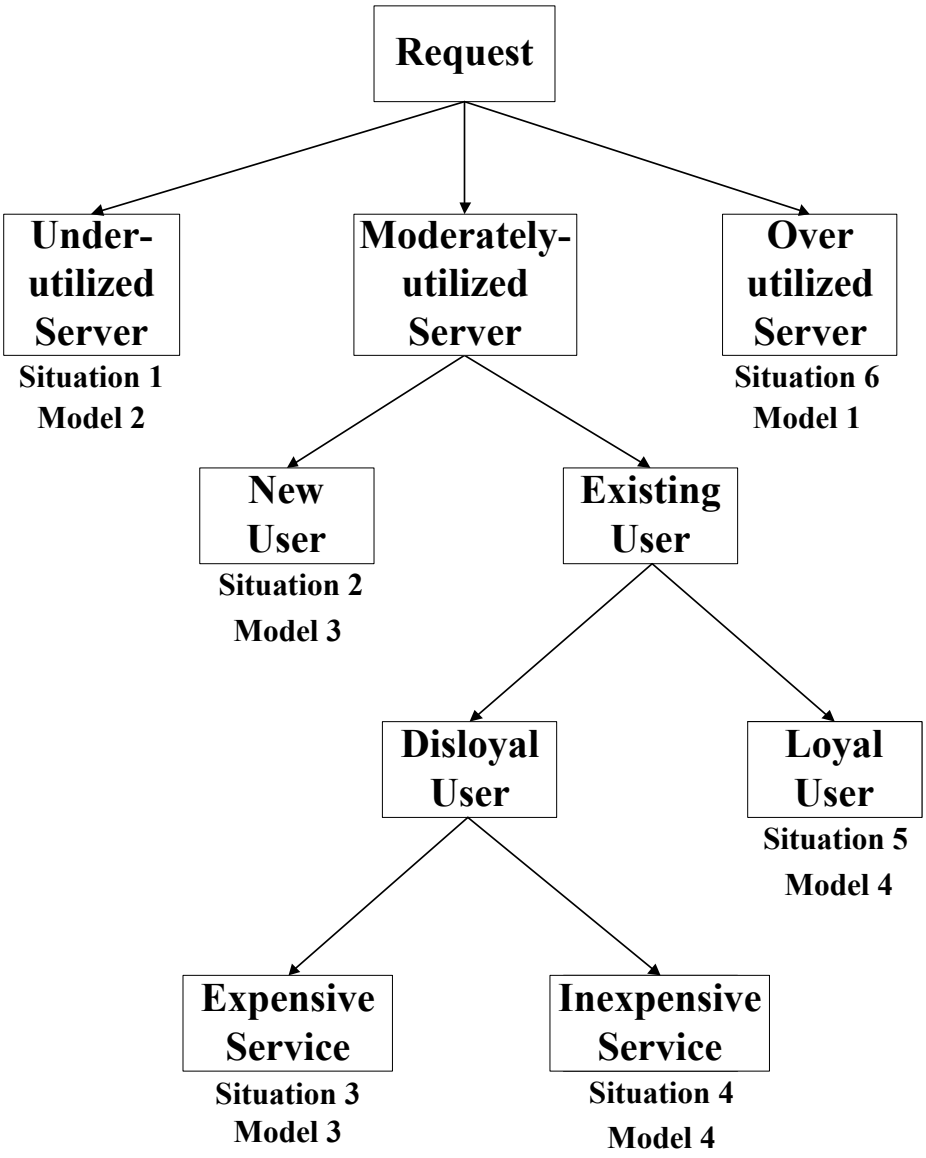

Figure 3.7: The decision tree of different situations

Situation 2: This situation happens when a first time customer is matched to a moderately-utilized server. As we know, first time users may have fluctuating needs which may affect their future services because their history is relatively short. According to the features of model 3, it will be a good choice for this situation. The value of $f_{(x)}$ does not change much when x is around 0.5, so the service performance will be maintained even if the customer's behavior is changeable. Also, the value of $f_{(x)}$ is neither high nor low while the overall amount of allocated resources and resource utilization in model 3 are both at medium level compared to other models, as Table 3.1 shows. This indicates that the provider will not suffer much loss even if the customer relinquishes most of what he receives.

Situation 3: This situation happens when an existing disloyal customer is matched to a moderately-utilized server, while the price of the requested service is expensive or the duration is long. In this situation, since the customer is disloyal and the server is moderately-utilized, the provider should not allocate too much resources. Considering the price and the duration, if the price of service is high or the duration of the service is long, then the service will be regarded as an expensive service. If the customer is always asking for resources from the same provider, it will be more beneficial for both the customer and the provider. Considering these points, the suitable model should be able to keep those customers by giving some "sweetener". Model 3 is a good fit since it treats disloyal users better than other models and it does not allocate too much resources which reduces the chance of heavy losses if those customers relinquish resources with high probability. Furthermore, model 3 gives good incentive for customers with really bad histories when they have better behaviors. It could be a good stimulation to let those customers to become more loyal.

Situation 4: This situation happens when an existing disloyal customer is matched to a moderately-utilized server, while the requested service is generally not expensive. Here the service is not expensive which means that the price is low or the duration is short or both. If the service is expensive but the duration is very short, it is not worth allocating a lot of resources since the provider would not make much money from those services because the customer is not loyal and the duration is short. It is the same when a service has a very low price but a long duration. In this case, model 4 is the most suitable model since it allocates fewer resources than other models except model 1. Compared to model 1, the strength of model 4 is that it gives better motivation to customers. As shown in Figure 3.4, when the customer is becoming better, the allocated resources increase fast. Compared to model 1, model 4 will have better performance in terms of increasing the server utilization and the profit for the provider. Besides, model 1 allocates the fewest resources among all the models, which is not necessary in a moderately-utilized server. According to these points, model 4 is the best model for situation 4.

Situation 5: This situation happens when a loyal existing customer is matched to a

moderately-utilized server. Since the server is moderately-utilized, resource utilization should be given more attention than in under-utilized situations. For customers, they should receive better services in this situation than in under-utilized situations. Besides, the request is submitted from a loyal existing customer, if more resources are allocated to loyal customers, the resource utilization will be higher. Combining these two points, model 2 and model 4 will be good choices since loyal customers can get much more resources in these models than in other models. In this thesis, model 4 will be chosen for this situation as model 4 allocates fewer resources to disloyal customers than model 2 does. Since in this situation the server is moderately-utilized, it can help the provider to serve more customers.

Situation 6: Situation 6 happens when a customer is matched to an over-utilized server. For a server which has few available resources, it is better to use them for the customers with the best records since the provider does not want to waste those resources. In this case, model 1 will be the best model for situation 6 because it allocates the fewest resources among all the models and the overall resource utilization of model 1 is the highest. As Table 3.2 indicates, in model 1, the amount of resources allocated to loyal users is seven times the amount of resources allocated to disloyal users.

When those models are different modules of one big model, it will have much better performance compared to the old model since the new model treats different situations with different suitable models while the old model treats different situations with the same strategy.

3.4 New Strategies of Resource Estimation for the First-time Cloud Service Customer

For cloud service providers, it is always good to attract more customers to use their services. It is always achieved by giving new users some "sweeteners". In [3], the model will have a default value of *SOP* or *AOP* for customers who are absolutely new or existing to the provider but request a service for the first time. After the first time, the model will do normal resource estimation based on customers' histories of past

activities. Since the record set is very short, it will be hard to make a reasonable estimation, which may cause a big fluctuation of the allocated services and hence influence customers' satisfaction. It is more important to keep customers loyal than just attracting customers. In this case, a new strategy of resource estimation is proposed to help make resource allocation more practical. It will use a slightly different strategy of resource estimation for the first three instances of a first time customer to attract and keep customers by optimizing the performance.

The new strategy divides the first time customers into two types: the ones having *AOP* records and the ones without *AOP* records. The broker will deal with both types of customers by following the steps outlined in Figure 3.8.

As Figure 3.8 shows, for first time customers, the value of *SOP* is set to 0.3. If this customer does not have any record for *AOP* either, set *AOP* to 0.3 as well. This strategy could attract more customers to use this service since 0.3 is regarded as a low relinquish probability, which was mentioned in [3]. Besides, in the second and third requests, the broker will do resource estimation with the average of the default relinquish probability and the real records. The default relinquish probability could reduce the fluctuation of allocated resources caused by the customer's fluctuating behaviors and also attract more users to enjoy the "opening offer". After the first three times, the client is treated as an existing customer and the broker will only use the actual records for regular resource estimation.

Algorithm 1 Resource estimation for new customers

```
1: procedure
2:     SOP: service oriented relinquish probabilities
3:     AOP: average overall relinquish probabilities
4:     if the user is a first time user then
5:         if the user has no record for AOP then
6:             SOP ← 0.3
7:             AOP ← 0.3
8:         else
9:             SOP ← 0.3
10:    else if the user is a second time user then
11:        SOP ← average of 0.3 and the record for the first time
12:    else if the user is a third time user then
13:        SOP ← average of 0.3 and the record for the first two times
```

Figure 3.8: The process for new customers

3.5 Pledge System

In the models introduced above, the broker will allocate services based on the records of cloud customers and their requests. Basically, the more loyal the customer is, the better service he will get. Similarly, customers with a bad history will get fewer resources for their requests. What if those customers really want a good service? To solve this problem, the pledge system is proposed to offer those customers a way to get the service they want.

In the pledge system, the broker will still do the resource estimation to get the amount of resources that would normally be allocated to the customer. Then, before doing the resource allocation, the broker will calculate the remaining part of the resources for the requested service, which is equal to the initial requested amount by the customer minus the result from the resource estimation. The amount of pledge the customer needs to pay equals to the cost of the remaining amount of resources plus 10 percent of overhead charge. Customers can get the service they want only if they pay for the part from the resource estimation and the pledge.

Another thing that is worth noting is the process of reimbursement. As we know, customers may relinquish the service before the service expires. In the models mentioned in previous sections, the broker will reimburse the customer according to the time left for the service. But if the customer decides to pay the pledge, the pledge

will not be reimbursed at all.

To make it easier to understand, let us look at the following example. Customer *A* with an overall average relinquish probability of 0.7 submits a request to the broker and the duration of the service is 10 hours. After the analysis, the broker knows that the service needs 100 GB of memory and 100 GB of storage. The price of memory is 0.1 dollar per GB per hour and the price of storage is 0.01 dollar per GB per hour. After the resource estimation using model 2 (Equation 13), the customer is supposed to get 51 GB of memory and 51 GB of storage and pay 56.1 dollars for ten hours (($0.01*51 + $0.1*51)*10). However, the customer wants to pay the pledge in order to get the rest of the resources (49 GB of memory and 49 GB of storage), the amount of the pledge is ($0.01*49 + $0.1*49)*(1+10%)*10 = 59.29 dollars for ten hours. So the customer will pay 56.1+59.29= 115.39 dollars to get the service for 10 hours. After 5 hours, if customer *A* decides to relinquish the service, he will only get a reimbursement of 56.1*5/10 = 28.05 dollars.

According to the pledge system, loyal customers will pay less for the complete service than disloyal customers (here the relinquishment of services and reimbursement are not considered since it is hard to predict the relinquish probability of the customer). The loss of the provider or the broker will be reduced a lot because the pledge will not be reimbursed in the pledge system. This strategy could stimulate customers to be more loyal and avoid unnecessary loss at the provider side. All types of customers can get their satisfied services by paying the pledge, which will not be reimbursed after customers relinquishing the allocated services.

3.6 The New Pricing Model

In the new model, because the customer has three types of resources to request: CPU, memory or storage, the pricing model will be built based on these three parameters. The price of each type of resources is estimated by the following equation:

$$\text{Price} = P_{CPU} * N_{CPU} + P_{Memory} * N_{Memory} + P_{Storage} * N_{Storage} \quad (20)$$

In Equation (20), P_{CPU}, P_{Memory} and $P_{Storage}$ stand for the prices to use a unit of resource per hour. For CPU, the unit is GHz while for memory and storage, the unit

is GB. N_{CPU} is the number of CPUs (GHz) to be allocated to the customer by the broker. Similarly, N_{Memory} is the size of the allocated memory (GB) and $N_{Storage}$ is the size of the allocated storage (GB). In the simulation part, the prices can be derived from some pricing models used by different cloud service companies, such as Amazon Web Service, Microsoft Windows Azure, etc.

Chapter 4

Simulation and Analysis of Results

In this chapter, we conduct simulations to evaluate the performance of the proposed models with an extensive range of scenarios. The tools used and the simulation environment are listed in Table 4.1.

Table 4.1: Simulation Setup

Operation System	Ubuntu 12.04 LTS
Memory	4 GB
Implementation Language	Java with Eclipse
Simulator	CloudSim 3.0.3

4.1 CloudSim

The simulation was done with the CloudSim cloud simulator. CloudSim is a framework for modeling and simulating cloud infrastructures and services [44]. CloudSim has many advantages: it can simulate many cloud entities such as data center, cloudlet and virtual machine. We can also build repeatable environments in CloudSim. The CloudSim structure is shown in Figure 4.1.

As Figure 4.1 indicates, we do not need to pay too much attention on the hardware and there are many features that we can add to the user coding area. For example, we can build new allocation methods in the class called “Data Center Broker”. Besides, we can create our own classes. In this simulation, we extend the framework by adding some new classes and new methods to make the environment of the simulation more complete and realistic. The important classes in our simulation and the relationships between them are described below and shown in Figure 4.2.

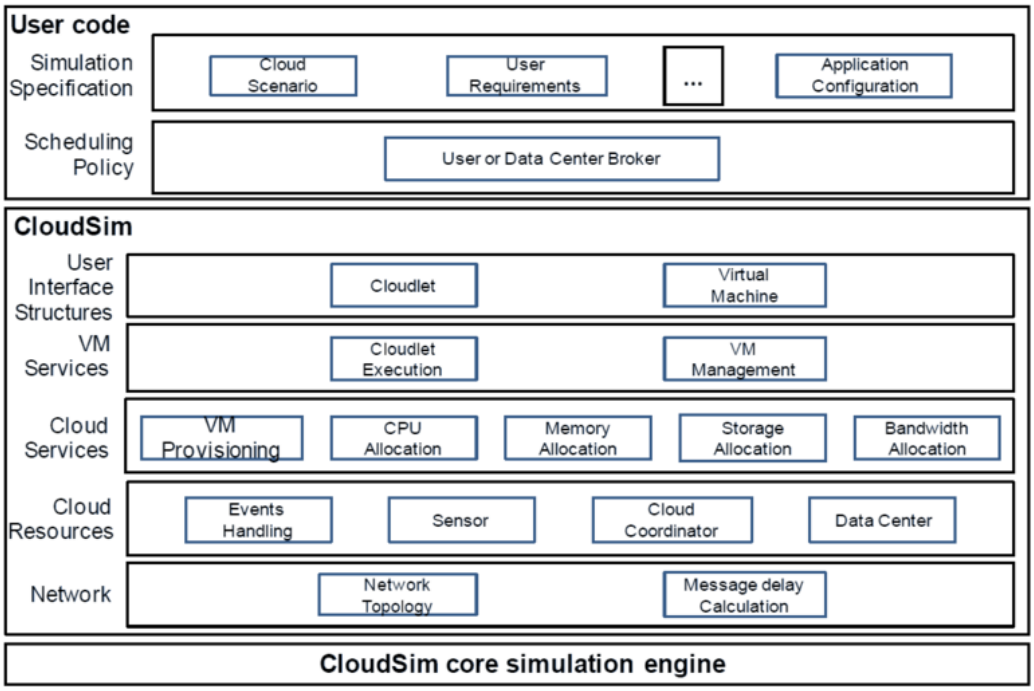

Figure 4.1: The structure of CloudSim[44]

Cloudlet: The cloudlet models the cloud-based application services (content delivery, business overflow) which are ordinarily deployed in data centers. For each application, it has parameters such as instruction length and the amount of data transfer which needs to be tackled for hosting the application successfully.

Datacenter: The "datacenter" class models the core hardware and software offered by the cloud providers. Each data center contains several hosts and each host has its own configuration of resources. Besides, a generalized resource provisioning component will be used by every data center component to implements policies for allocating resources.

Data center broker: The data center broker is responsible for the negotiation between the user (i.e. cloud customer) and the host (i.e. cloud provider) for the allocation of resources and doing the match making between virtual machines and cloudlets. It needs to be extended for conducting simulations with customized allocation policies.

Host: The host models a physical server. Each host contains several virtual machines and includes some important information such as the amount of memory and storage, the record of each user and the allocation policies for sharing and provisioning

resources among virtual machines (time-shared allocation policy, space shared-allocation policy). It is also extended to keep the history of each user.

User: The user models a cloud service customer. A user may request for several cloudlets. A data center broker will analyze each request of a user and do the match making between users and cloud providers. Related hosts will keep the history of each user afterwards.

Virtual Machine: A virtual machine runs on a host to process Cloudlets [16]. Every virtual machine has access to get resources like memory storage and processor from the host according to the allocation policy.

As Figure 4.2 indicates, the Cloud Information Service is an entity that provides services including indexing, discovery and resource registration. It is created in CloudSim during the initiation of the simulation. The data center models the core infrastructure-level services that are offered by cloud providers like Amazon, Azure, etc. Each data center contains several hosts. Each host has several virtual machines and the policy for sharing resources among virtual machines. On the other side, users submit their requests to the data center broker. Each user has several Cloudlets to be processed. The data center broker analyzes those requests and finds the most suitable resources for each user. Each virtual machine can process one or more Cloudlets.

The flow chart of the simulation is shown in Figure 4.3. First of all, the main elements of CloudSim will be generated such as users (Cloud Service Customer), data center broker, host (Cloud Service Provider), etc. Then the relationships between those classes are built. After that, the customer sends a request to the data center broker which will be analyzed. The broker gets the necessary amount of resources for this request, calculates how much resources should be allocated according to the resource estimation model and then buys the resource from the suitable provider. Afterwards, the data center broker calculates a price for the resources allocated to the customer. As we can see, the user pays the upfront fees before the task is processed. If the customer relinquishes some resources, the data center broker will calculate how much should be reimbursed to the user. At the same time, the provider will keep the record for this customer. The reason that the customer pays beforehand is to avoid

cases where remote customers will not pay after they have used the resources. For instance, a customer in Canada may use Amazon Web Service in Virginia and refuses to pay after using the service.

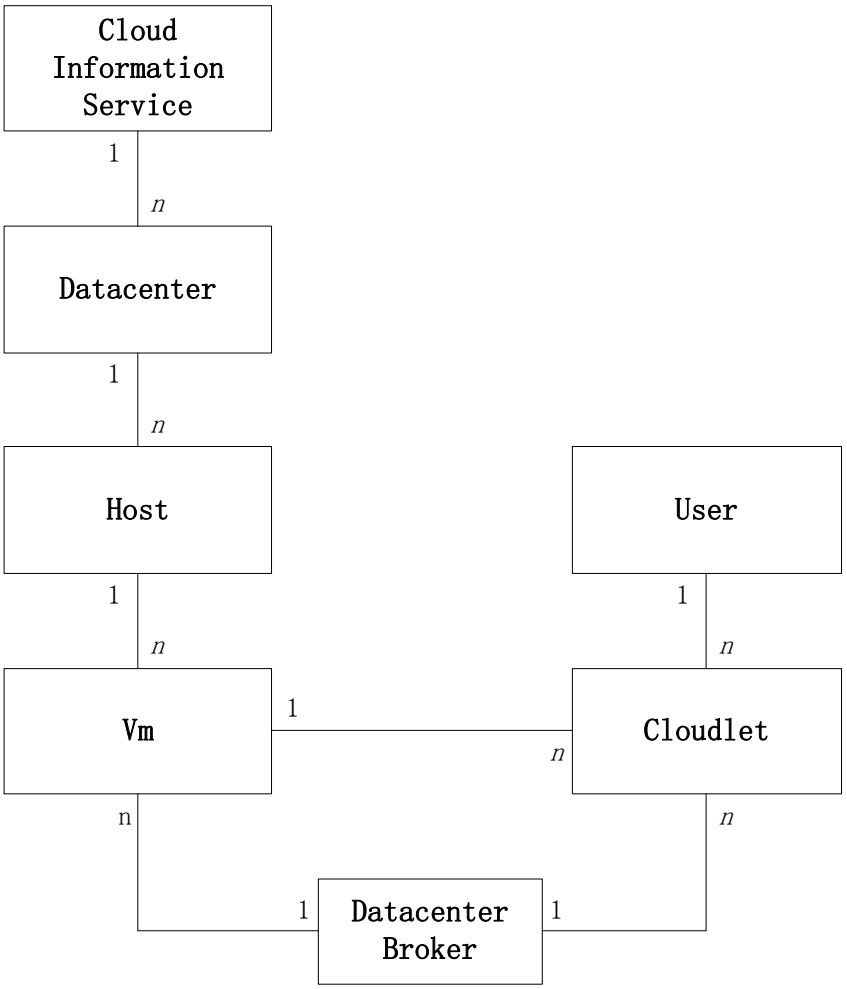

Figure 4.2: Relationship between the main classes in CloudSim

This kind of situation can be very tricky to deal with. So the strategy shown in Figure 4.3 can help the provider avoid things like this. It is worth noting that this flow chart represents the simulation of a single service. However, it can be repeated if one customer asks for more than one service from the same provider, or more than one cloud service customers ask for services from more than one cloud service providers.

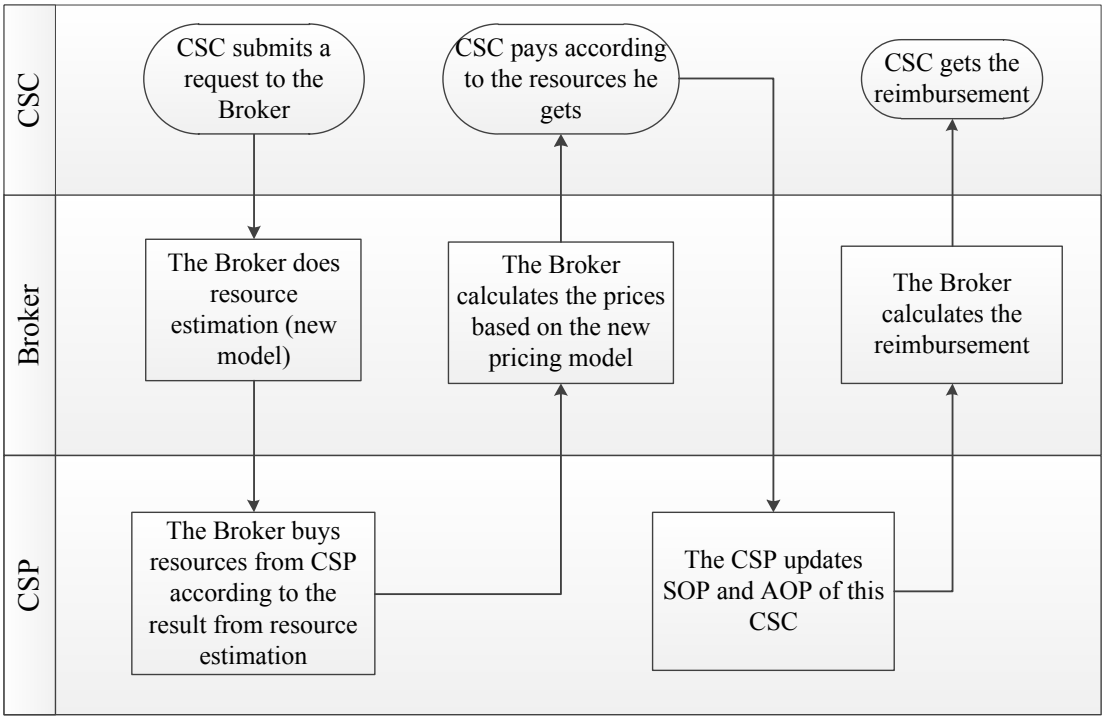

Figure 4.3: Flow chart of the simulation

When the data center broker receives a request from a customer and has already found the most suitable provider for the request, the combined model (i.e. the new model proposed in this thesis) will use different modules for different scenarios. With different modules, the new model can allocate resources in a more reasonable way for different types of customers and scenarios; thereby increase the overall server utilization of cloud service providers.

4.2 The User's Behavior

The Gaussian distribution is used to emulate the behavior of the customer. For example, to simulate a disloyal customer with an average relinquish probability of 0.7, the following Gaussian distribution is used to generate relinquish probabilities for this customer:

$$X \sim N(0.7, 0.3) \tag{21}$$

As shown above, the mean is 0.7 and the standard deviation is 0.3, representing that the average relinquish probability of this customer is 0.7. According to the definition of the Gaussian distribution, 70% of the instances will tend to have a

value one standard deviation on either side of the average. In other words, 70% of the values generated from the distribution model will be between 0.7 +/- 0.3. 95% of the values will be between 0.7 +/- 0.6 and more than 99% of the values will be between 0.7 +/- 0.9. In the simulation part, there will be a filter to discard the values which are lower than 0 or larger than 1. Based on this, we can use those generated and filtered relinquish probabilities for simulation. It is worth noting that the value of the standard deviation could be changed for different purposes.

4.3 A Single Customer

In this section, we will run a simple simulation for a single user to introduce the whole process. In this simulation, a first time customer will request the same service 10 different times with an overall average relinquish probability of 0.5. Table 4.2 lists the values of the key parameters in this simulation.

It is worth noting that the relinquish probabilities listed in Table 4.3 are generated by the Gaussian distribution where the average value is set to 0.5 and variation is set to 0.3. For each instance, the customer will relinquish resources according to the relinquish probabilities in Table 4.3. In this example, the data center broker will do resource estimation using model 1. The pricing model introduced in the last chapter is used for this simulation, which is derived from the Amazon pricing model [24].

Table 4.2: Input Setup for One Customer

Parameters	**Value**
Requested CPU (GHz)	300
Requested memory (GB)	300
Requested storage (GB)	300
P_{CPU} (dollar)	1/100
P_{Memory} (dollar)	1/100
$P_{Storage}$ (dollar)	1/1000
Model	Model 1
Duration (h)	200

Table 4.3: Details of Important Parameters for Each Instance

CSC ID	Instance No.	Resources Requested	SOP	AOP	$f_{(x)}$	Relinquish Probability
1	1	300	0.3	0.3	0.49	0.5315137618
1	2	300	0.4157568809	0.5315137618	0.2977420987	0.3887194266
1	3	300	0.4067443961	0.4601165942	0.331159818	0.3545787098
1	4	300	0.4249372994	0.407347652	0.3374749203	0.7264482387
1	5	300	0.5003150342	0.5756927691	0.2252062959	0.2514931734
1	6	300	0.4505506621	0.3759041038	0.3298566995	0.5728758639
1	7	300	0.4709381957	0.511713263	0.2657094397	0.8990069165
1	8	300	0.5320908701	0.6849725561	0.1738461083	0.507646189
1	9	300	0.529035285	0.5198685296	0.224695245	0.1559978144
1	10	300	0.4875866771	0.3425165497	0.3144630288	0.4517433661

Table 4.3 shows all the details and especially how the amount of allocated resources for each instance is affected by the customer's behavior. At the beginning, since this customer is a first time user, the data center broker will implement the strategy for new users and use the default value of 0.3 for the SOP and AOP of this customer to do resource estimation. After the first three instances, the data center broker will switch to the normal strategy for resource estimation. For each instance, after calculating the new SOP and AOP, the overall relinquish probability x will be calculated by Equation (10) and then be used for resource estimation. Then, the broker will allocate resources based on the result of $f(x)$ calculated by Equation (12). If the customer has a bad behavior for one instance, the broker will do the reimbursement and the cloud service provider will keep the record. The record will have an influence on the overall history of the customer and therefore affect the result of the resource estimation and the amount of resources allocated afterwards.

According to Figure 4.4, we can see how the resource estimation is affected by user's behaviors in a more explicit way. The blue line represents the relinquish probability for each instance (as indicated in the last column of Table 4.3) and the red bars represent the amount of the allocated resource. In Figure 4.4, CPU is used to

represent the general allocated resources. At the fourth, the sixth and the seventh instances, the user relinquished resources at an obviously higher probability than the previous instances. Combining the statistics from Table 4.3, the relinquish probabilities in those instances are higher than the average relinquish probabilities calculated from the previous records. Therefore, the value of $f_{(x)}$ is less in the next instance which causes a drop in the allocated resources. On the other hand, at the fifth, eighth and ninth instances, the user's behaviors are much better than the former behavior and the relinquish probabilities are also lower than the average, so the allocated resources rise up in the next instance.

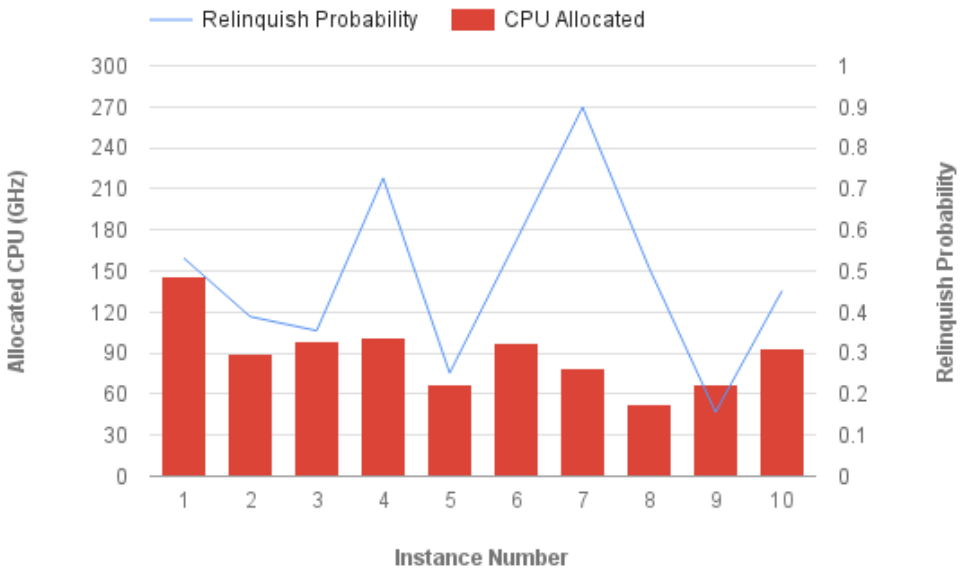

Figure 4.4: Relinquish probability vs allocated resources for a single customer

4.4 Existing Customers vs. First Time Customers

In this section, a simulation is designed and implemented to find out how differently will the broker treat new customers and existing customers. As we know, the broker implements a different strategy for first time customers. With the strategy for first time customers, the fluctuation of the amount of allocated resources will be reduced and will be more stable as more and more records are collected. In this case, simulations are implemented with two groups of customers, one with 20 existing

customers and the other with 20 first time customers. To create the record of the existing customers, we generated 20 instances using the Gaussian distribution discussed earlier with a mean of 0.5 and a variance of 0.3. In this simulation, each customer (existing and new) is asking for the same service 10 times with an average relinquish probability of 0.5, also generated by Gaussian distribution. Both types of customers will relinquish services using the same probabilities generated by Gaussian distribution during the simulation. Besides, both types of customers will request for the same services (i.e. they are requesting the same amount of resources). For each instance, the customer will ask for 300 gigahertz of CPU, 300 gigabytes of memory and 300 gigabytes of storage from the cloud service provider. Similar to the last section, the data center broker will use model 1 to estimate the amount of resources. The pricing model is derived from the Amazon pricing model [24]. The important parameters are shown in Table 4.4. It is implemented the same way for each customer.

Table 4.4: Input Parameters for Both Groups of Customers

Parameters	**Value**
Requested CPU (GHz)	300
Requested memory (GB)	300
Requested storage (GB)	300
P_{CPU} (dollar)	1/100
P_{Memory} (dollar)	1/100
$P_{Storage}$ (dollar)	1/1000
Resource allocation model	Model 1
Duration (h)	200

Since the two groups of customers are using the same generated relinquish probabilities, we can compare the results of the resource estimation and evaluate the impact on these two groups. The amount of resources allocated to each customer is collected and then the average for each group of customers is calculated and used for comparison. As Figure 4.5 shows, the orange and the red bars stand for the average amount of the allocated resources of 20 first time customers and 20 existing

customers respectively. Similarly, the blue line represents the average of relinquish probabilities. As we can see, at the beginning of the simulation, the variation in the amount of allocated resources to new users is larger than to existing customers. This is because the broker uses a different strategy to estimate resources for first time customers. With this strategy, the broker will assign them more resources in first three instances. However, the relinquish probabilities are still stored by the provider in order to start building the customer profile. From the fourth instance, the difference between the allocated resources for new and existing customers is gradually stabilizing.

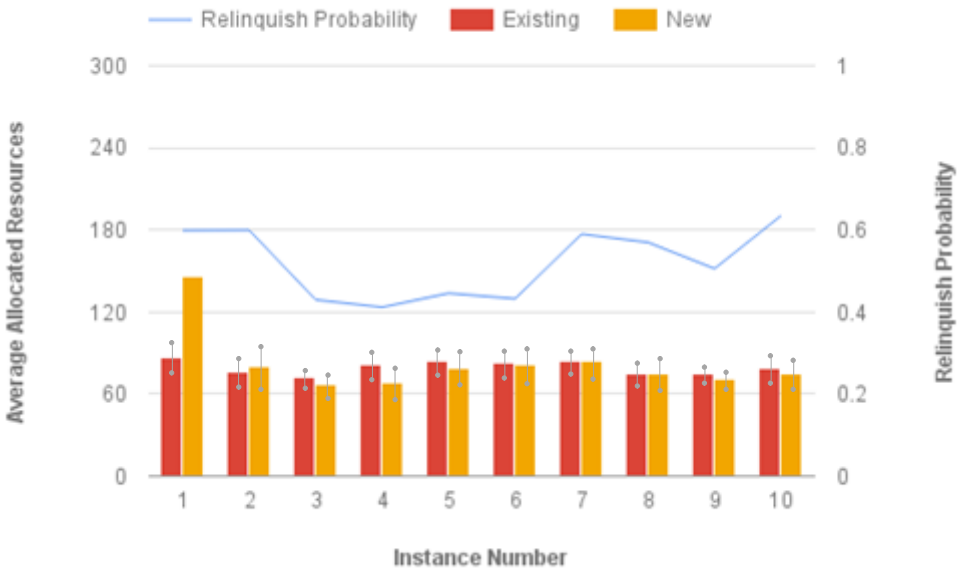

Figure 4.5: Relinquish probability vs allocated resources for existing users and new users

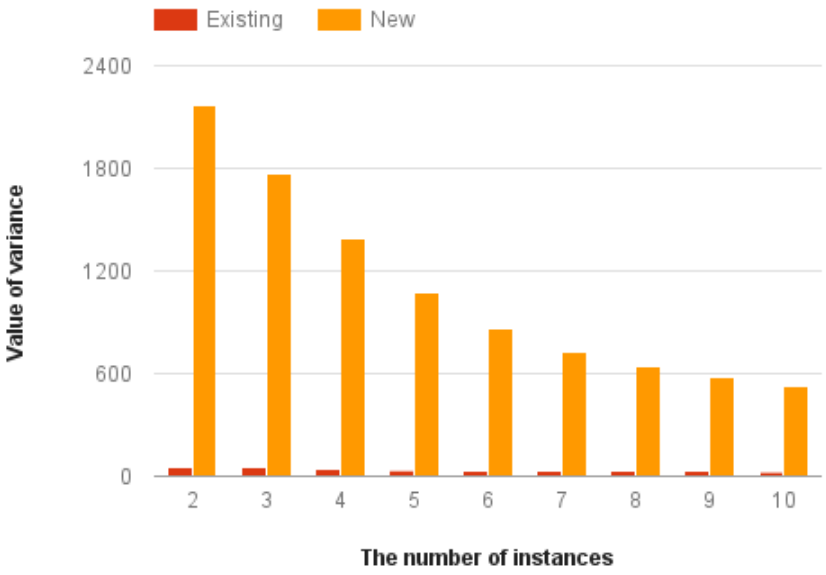

Figure 4.6: The trend of variance of allocated resources for both types of customers

Figure 4.6 gives us more details about how the variance of the amount of allocated resources changes for both types of customers during the simulation. For each customer, variances of allocated resources at different stages during the simulation are calculated. The red bars stand for the average of variances of existing customers while the yellow bars represent the first time customers. For each type of customers, the variance is the average of 20 customers. The instance number starts from 2 since at least two numbers are needed to compute the variance. Since all existing customers have records of 20 instances and the services allocated to existing users are stable, the variance is much lower than the variance of allocated resources for first time users. On the other side, the variance for the first time customers is very big at the beginning but it gradually decreases as more records are obtained. It can be predicted that as time goes by, the variances of two types of customers will be near each other as both of them have more records.

4.5 Comparison of Each Model from Simulation Results

In the previous chapter, the proposed models were introduced for different situations that can happen in a cloud computing environment. Their performances were

analyzed mathematically, including the overall amount of allocated resources; the overall server utilization and how they allocate resources according to customers with different average relinquish probabilities. According to that, the suitable scenarios for each model were discussed. In this section, these models will be used for resource estimation in CloudSim. Their performances will be analyzed and compared with the mathematical performances obtained in the previous chapter.

Table 4.5 lists all the parameters used for each model. There will be eleven groups of customers with different average relinquish probabilities from 0, 0.1, 0.2 to 1. They will be asking for the same service from the same cloud service provider. Each group has ten customers and they will relinquish services on probabilities generated by Gaussian distribution. The mean of the Gaussian distribution is the average relinquish probability of each customer and the standard deviation is 0.3. It is worth noting that all customers are existing customers because that will avoid the unnecessary fluctuation caused by first time customers. During the simulation, some important outputs are collected such as the amount of allocated resources, the overall payment and the amount of money the provider earns from the customer (the payment minus the reimbursement). They will be used for comparison with the mathematical performance. In order to calculate the server utilization, it is assumed that the provider has 300 gigahertz of CPU, 300 gigabytes of RAM and memory, which is same as the request of each customer. This means that if the customer gets the service he wants, the server utilization will be 100%, which is the same as the mathematical analysis.

Figures 4.7 to 4.10 show the comparison between the mathematical performances discussed in the previous chapter (red lines) and the performance from the simulation of each model (blue lines). For each value of the relinquish probability, the amount of allocated resources is actually the average over 10 requests from different customers (all having the same mean and variance). After that, we have to divide these numbers by the total amount of requested resources. This is done so that we can have two sets of comparable data. Recall that in the last chapter, the mathematical analysis was done with the assumption that the total amount of requested resources was 1.

Table 4.5: Input Parameters for All the Models

Parameters	**Value**
Requested CPU (GHz)	300
Requested memory (GB)	300
Requested storage (GB)	300
P_{CPU} (dollar)	1/100
P_{Memory} (dollar)	1/100
$P_{Storage}$ (dollar)	1/1000
Duration (h)	200

As Figure 4.7 indicates, the red lines are similar to the blue lines, especially for the middle part while x is around 0.5. When x is near 0 or 1, we can clearly see some deviations in the four graphs. This is because during the simulation, the relinquish probabilities used by the customers are randomly generated by Gaussian distribution with a mean equal to the average relinquish probability of each customer and a standard deviation equal to 0.3. For instance, if the average relinquish probability of customer *A* is 0.9, the Gaussian distribution will use 0.9 as its mean and 0.3 as its standard deviation to generate the relinquish probabilities. According to the Gaussian distribution, 70 percent of the generated relinquish probabilities will tend to have a value within one standard deviation (which is 0.3 here) on either side of the mean (in this case, values between 0.6 and 1.2 but after the filtering it will be between 0.6 and 1). Furthermore, around 99% of the instances will tend to have a value within three standard deviations on either side of the mean (between 0 and 1.8 in this case and filtered to values between 0 and 1). Therefore, this process may generate relinquish probabilities that are far from the average, which in turns may affect the graphs by rising or dropping slowly from the middle to the edge (e.g., model 3, the left part of model 1 and the right part of model 2). However, in the mathematical model, it is assumed that every customer will relinquish resources on the probability equal to its own average relinquish probability. That explains the difference between the

mathematical graph and the simulation graph, especially when x is getting closer to 0 and 1. The reason why the standard deviation is set to 0.3 is to imitate realistic situations. In practice, it is hard to predict customer behavior - even a trustworthy customer may have some bad records due to unpredictable events, and vice versa.

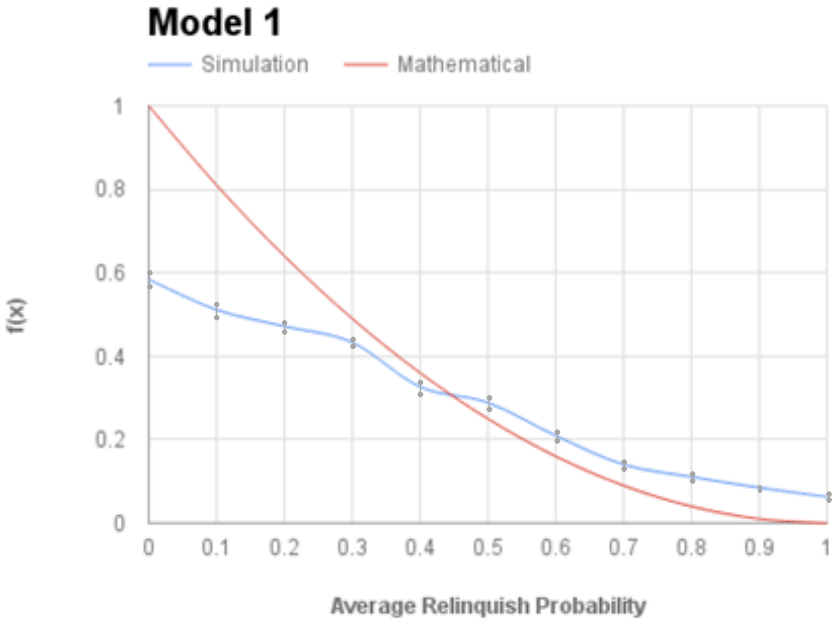

Figure 4.7: Comparison between the mathematical model and the simulation - model 1

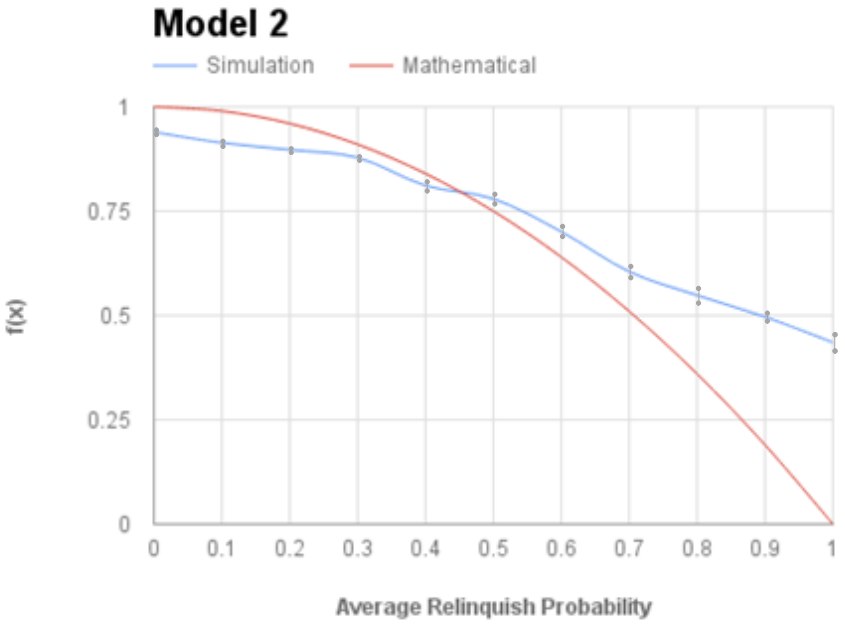

Figure 4.8: Comparison between the mathematical model and the simulation - model 2

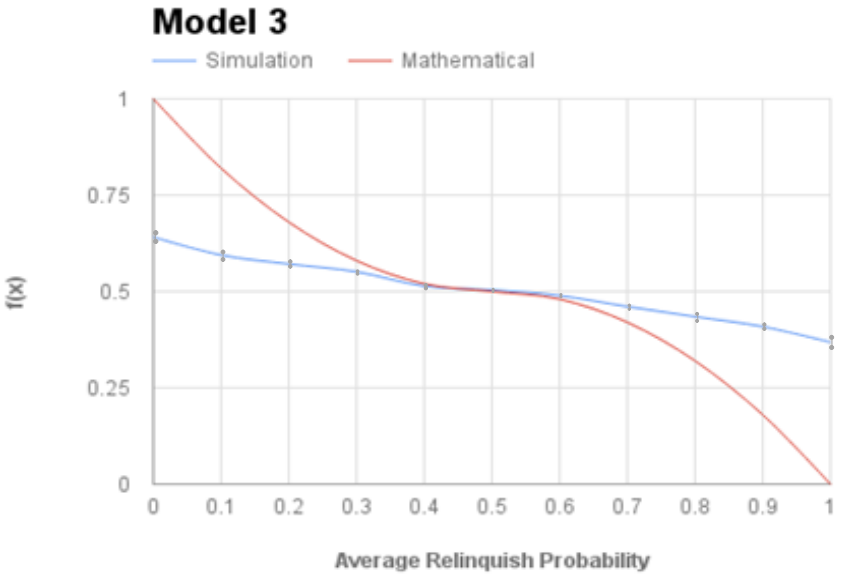

Figure 4.9: Comparison between the mathematical model and the simulation - model 3

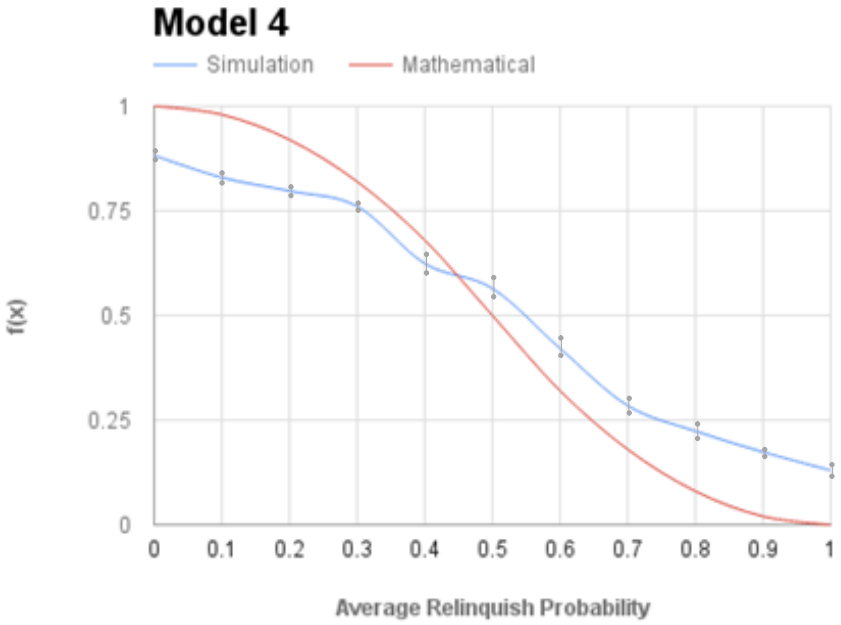

Figure 4.10: Comparison between the mathematical model and the simulation - model 4

From figures 4.7 to 4.10, the models in the simulation treat different types of customers the same way as the mathematical models. As mentioned before, the standard deviation of the Gaussian distribution is set to 0.3 which may affect the graph which rises or drops slowly from the middle to the edge, but for other parts of these graphs, the performance is close to the mathematical performance. Apart from

that, the general shape and trend of the blue lines are very close to the red lines in each graph. In summary, it can be concluded that the proposed models work the same way in the simulation as how they work in an ideal way (i.e. mathematical way). If a customer is always a loyal customer for the cloud service provider, according to the proposed model, he will always be able to receive stable and high-quality services.

To make the comparison more complete, the sever utilization is also regarded as a metric for comparison between the simulation results and the mathematical results. Figure 4.11 shows the comparison of the server utilization calculated from the mathematical models and the simulation results. The server utilization is calculated by Equation (21). For each model, the average amount of allocated resources for all groups of customers is used to represent the amount of allocated resources. 300 is the total amount of resources that the cloud service provider holds. Then, the server utilization for each model is calculated by dividing the amount of allocated resources by the total amount of resources.

$$Utilization = \frac{allocated\ resources}{300} \tag{21}$$

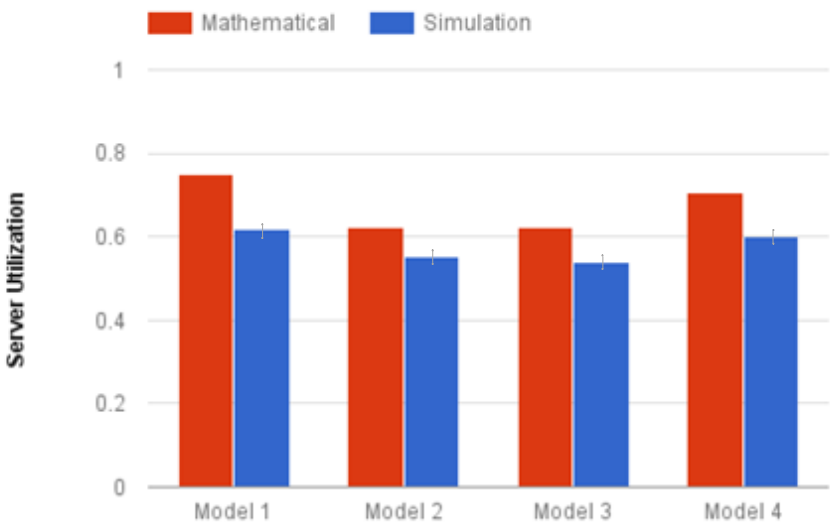

Figure 4.11: Comparison of server utilization of proposed models

As we can see, the red bar is higher than the blue bar for each model, which indicates that the server utilization of the mathematical model is higher than the server utilization calculated from the simulation results. The main reason is the same as the one explained in last section as the relinquish probability is fluctuating according to the Gaussian distribution. Therefore, as Figure 4.11 appears, the model will allocate more resources to disloyal customers or allocate fewer resources to loyal customers in different levels, which will cause the decrease of the server utilization. But generally, the difference is not much as we can see from the figure. Besides, the relationship among the server utilizations from the simulation results is same as the relationship among the mathematical server utilizations. Model 1 has the highest server utilization, followed by model 4, while the bars standing for model 2 and model 3 are pretty much the same height. In conclusion, even if the simulation is not an ideal environment, as for server utilization, the proposed models perform the same way as they perform under ideal conditions.

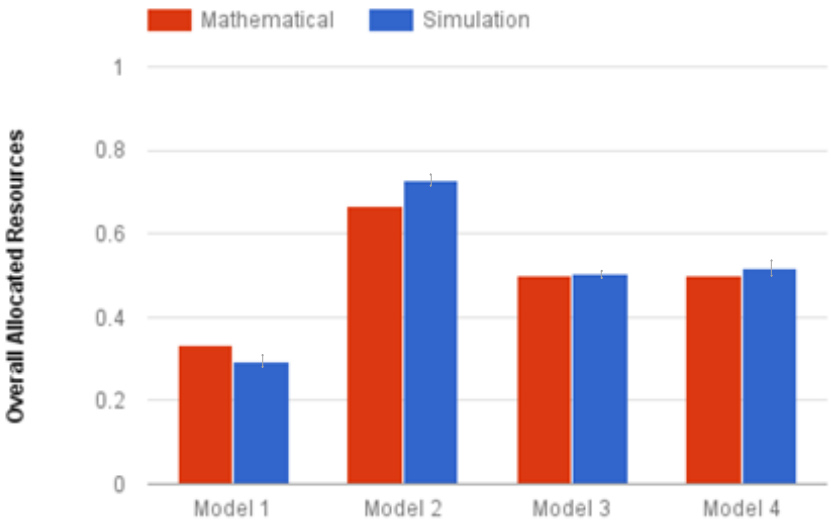

Figure 4.12: Comparison of overall allocated resources of proposed models

Figure 4.12 gives us the comparison of the overall amount of allocated resources

for each proposed model. Since the amount of requested resources was set to 1 for the mathematical models, we have to divide the simulation results by the total amount of requested resources in order to have a fair comparison. As we can see, model 1 allocates a bit fewer resources in simulation as shown in Figure 4.12, while other models generally allocate a little more resources to customers in the simulation. Basically, we can say that there is not much difference between the mathematical results and the simulation results.

4.6 The Pledge System

In this section, simulations are designed to find out how the pledge system will affect the performance of the resource estimation. As introduced in Chapter 3, the pledge system will allow disloyal customers to receive the service they want and at the same time, providers will not lose more money if customers relinquish the service. The expected result of this simulation is that when a customer gets the service with the pledge, the reimbursement will not increase and the provider will also earn more from the pledge.

All of the four models will be implemented in this simulation. There are several groups of customers with different average relinquish probabilities in this simulation, and all customers in this simulation are existing customers with a history of 35 records.

Table 4.6 lists the key parameters that each model will use during the simulation. For each model, five groups of customers will request for the same service. Their average relinquish probabilities and the information of the requested services are listed in Table 4.6. The duration of each request is 10 hours. The pricing model is the same as the pricing model discussed in the last section.

In the simulation, resource estimation with and without pledge is performed for each customer. Then, the allocated resources, the amount of reimbursement and the income are collected. For each model, the average data of the five groups of customers will be used for comparison.

Table 4.6: Input Parameters for testing the pledge system

Parameters	**Value**
Requested CPU (GHz)	300
Requested memory (GB)	300
Requested storage(GB)	300
P_{CPU} (dollar)	1/100
P_{Memory} (dollar)	1/100
$P_{Storage}$ (dollar)	1/1000
Duration (h)	10
Relinquish Probability	0.1, 0.3, 0.5, 0.7, 0.9

As Figure 4.13 indicates, for each model, the broker will allocate the requested amount of resources when the pledge system is used. More importantly, the reimbursement does not change much as Figure 4.14 shows. This means that cloud service providers will not lose more money if the broker utilizes the pledge system and allocates more resources to disloyal customers because the pledge part will not be reimbursed if the customer relinquishes the service. For example, the broker would allocates 80 GB RAM to a customer and get 80 dollars (suppose the price for RAM is 1 dollar per GB) without the pledge system. But with pledge system, the broker will allocate 200 RAM to this customer and the customer needs to pay for the extra resources plus 10 percent of the extra payment for the pledge. Then, if the customer relinquishes the service, the extra payment will not be reimbursed. Therefore, the cloud provider will not lose more money if the pledge system is implemented. In Figure 4.15, it also matches the expectation that the pledge system could help the providers make more money by allocating more resources to disloyal customers. Since all the models work very well under the pledge system, the combined model will also work well using the pledge system for different situations.

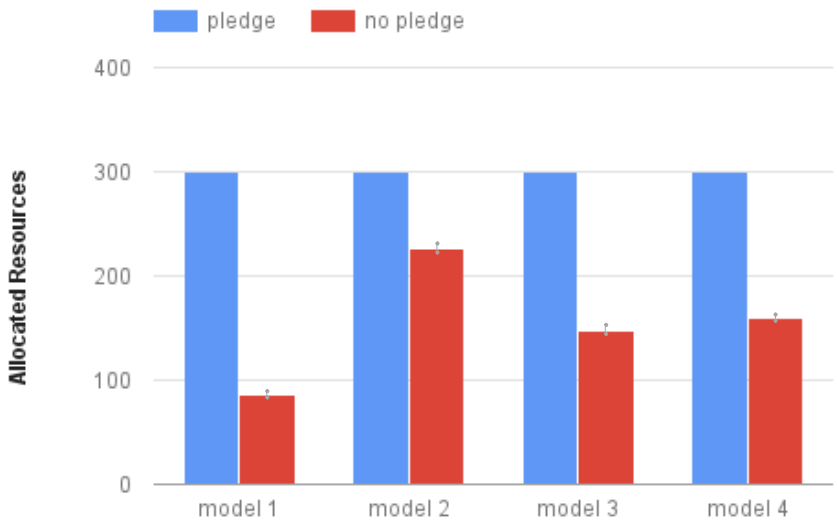

Figure 4.13: Comparison of allocated resources of pledge system and non-pledge system

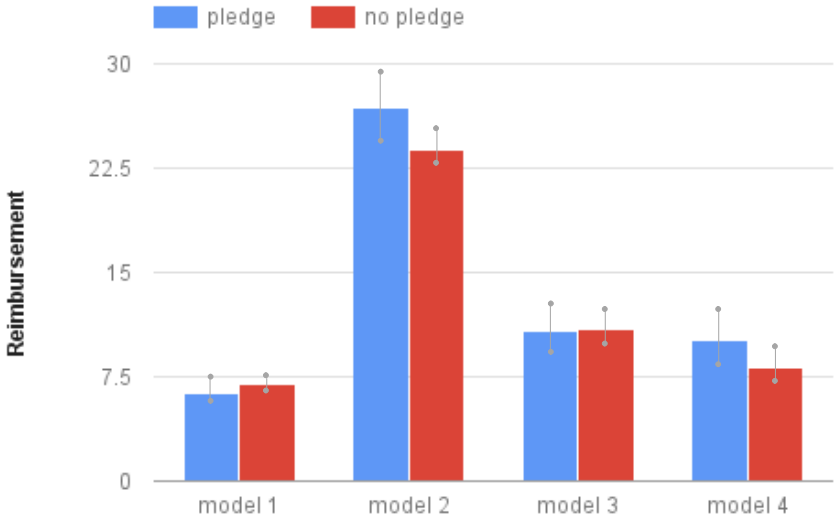

Figure 4.14: Comparison of reimbursement of pledge system and non-pledge system

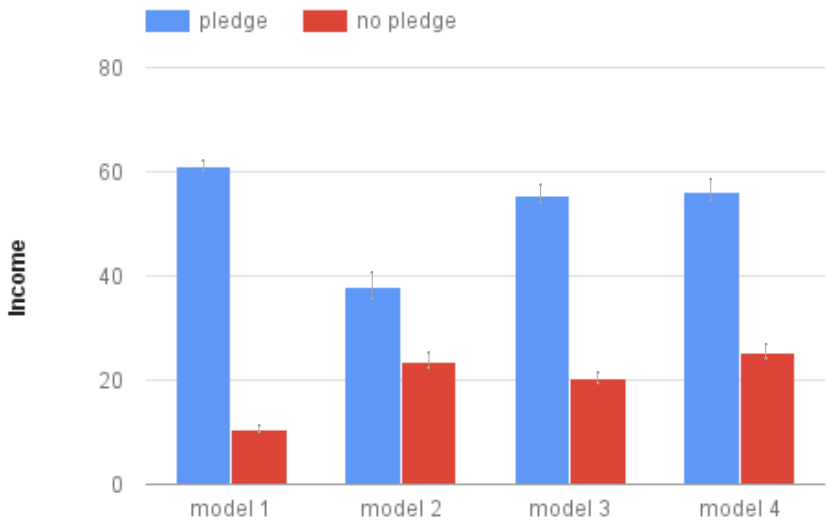

Figure 4.15: Comparison of the income of pledge system and non-pledge system

4.7 The Combined Model vs. The Old Model

In [3] and as also mentioned in the last chapter, a single model is deployed in the data center broker and that model is used for all possible situations in a cloud computing environment. Although it reduces the burden of the broker, it will not provide the best performance in real life since there are many complicated scenarios and obviously one model is not enough to deal with them. Therefore, a model comprised of all the proposed models is deployed in this simulation to improve the overall performance of the cloud providers. In this model, the broker will utilize different modules according to the corresponding scenarios as discussed in the last chapter.

To compare the combined model with the old model, we decided to compare the performances of both models over a time period of 100 hours. During this time period, there will be different types of customers asking for different services for different duration. With the new model, the broker will decide which model to use for each request according to several metrics such as the server utilization at that time, the type of each customer and the information of the requested service as shown in Figure 3.7. CloudSim is extended for this simulation as the system will update the server utilization during the simulation. It will automatically recalculate the server utilization

as some resources are allocated or some services are expired and keep the record with the time stamp. The key input parameters are listed in Table 4.7 for 20 requests. The time stamp is the time when the customer submits the request to the data center broker and the amount of requested resources stands for the amount of each type of resources. For example, the amount of requested resources for the first instance is 100. This means that the customer asks for 100 GHz of CPU, 100 GB of memory and 100 GB of RAM from the cloud service provider. The duration of the requested service represents how long the customer will use the service from the time he submits his request. It is worth noting that the actual duration of the service will be calculated during the simulation using the relinquish probability for this service which is generated by the Gaussian distribution. The mean of the Gaussian distribution is the average relinquish probability of each customer and the standard deviation is set to 0.3. The average relinquish probability for each customer is also listed in Table 4.7 and for first time customers, the data center broker will use default values (0.3 as mentioned in Section 3.4) for the average relinquish probabilities.

During the simulation, the time stamp and the corresponding server utilization is collected and it will be fluctuating as some resources are allocated or some services are expired. To eliminate the deviation, the simulation will use the same input parameters running for 10 times to get 10 sets of data for both models. The average number is then used to draw the line of the server utilization for both models. With the trend of the server utilization, we can precisely observe how the data center broker works under the combined model and the old model and figure out if the combined model provides better performance by increasing the overall server utilization compared to the old model.

Table 4.7: Input parameters

Request Number	Time Stamp (h)	Amount of Requested Resources	Duration of the Service (h)	Average Relinquish Probability of the Customer
1	0	100	192	0.5
2	3	50	50	0.69
3	5	100	96	first time user
4	10	200	60	0.38
5	14	50	100	first time user
6	23	300	56	0.53
7	30	50	80	0.76
8	32	200	70	0.58
9	36	250	84	0.45
10	38	150	96	0.28
11	44	66	63	0.36
12	46	80	54	0.82
13	50	300	60	0.55
14	54	150	72	0.86
15	70	300	50	first time user
16	84	300	60	0.6
17	86	50	50	0.3
18	89	200	48	0.74
19	91	100	50	0.22
20	99	55	10	0.49

Figure 4.16 shows the server utilization for both models over a time period of 100 hours. It is worth noting that the confidential intervals are really small (less than 1%) so they are not shown in the figure. As we can see, from the beginning of the simulation to the end, the performance of the combined model is always better than the old model. While in the new model the provider is over-utilized (i.e. server utilization greater than 75%), the provider using the old model still has nearly half of the resources remaining unutilized. Under the combined model, the server utilization of the provider is higher than 50 percent for nearly half of the simulation time. On the

other hand, the server utilization of the provider using the old model scarcely exceeds 50 percent. In other words, the combined model performs much better than the old model with respect to increasing the server utilization by treating different scenarios with different modules. In this simulation, all the situations mentioned in the decision tree are included (all kinds of customers, requests and server utilizations are included, as Figure 4.16 and Table 4.8 shows). Therefore it is proved that the new model performs well under every situation as expected.

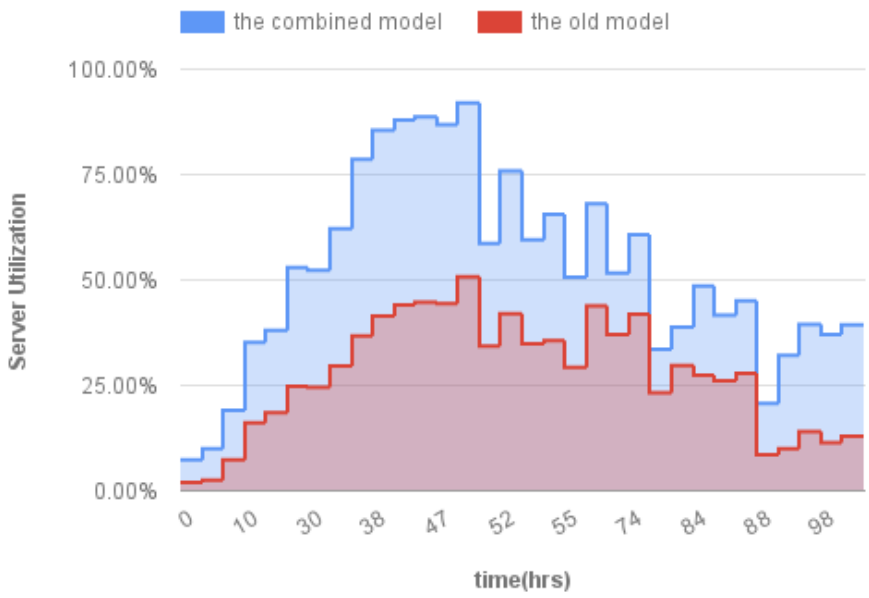

Figure 4.16: Server utilization in the simulation for both models

In the previous real-time simulation, the parameters of the 20 requests were fixed. In order to evaluate the system, we run a similar simulation where the duration, the arrival time, the relinquish probabilities and the amount of requested resources are randomly generated. It is assumed that customers will submit request based on the M/M/1 queuing model so the inter-arrival times are generated using the exponential distribution with the mean equal to 4. Relinquish probabilities are still generated by the Gaussian distribution, where the mean is randomly generated between 0 and 1 and the variance is set to 0.3. The basic input parameters are listed in Table 4.8. For each simulation, the inputs are the same for both models and the average server utilization

under each model is collected for comparison. This simulation setup is run 20 times in order to get reliable results. As we can see from Table 4.9, for all simulations, the new model performs better than the old model. In those simulations, the server utilization under the new model is about two times the server utilization of the old model. On average, the difference of the server utilizations between the two models is 18.07%. The 95% confidential interval of the difference is 2.7%.

Table 4.8: Input Parameters for real-time simulation

Parameters	**Value**
Range of requested resources	(0,200]
Range of durations	(20,50]
P_{CPU} (dollar)	1/100
P_{Memory} (dollar)	1/100
$P_{Storage}$ (dollar)	1/1000
CPU in the cloud (GHz)	1000
Memory in the cloud (GB)	1000
RAM in the cloud (GB)	1000
Duration of the simulation (hrs)	100

Table 4.9: Average server utilization of simulations with random inputs

Simulation No.	Average Server Utilization (new model)	Average Server Utilization (old model)	Difference
1	25.81%	12.33%	13.48%
2	30.38%	13.46%	16.92%
3	20.49%	9.34%	11.14%
4	37.24%	20.17%	17.07%
5	52.01%	27.43%	24.58%
6	25.74%	11.38%	14.37%
7	27.35%	12.70%	14.65%
8	21.51%	10.24%	11.27%
9	29.31%	13.75%	15.56%
10	22.78%	10.53%	12.25%
11	27.73%	14.35%	13.38%
12	56.02%	29.23%	26.79%
13	47.74%	28.58%	19.16%
14	58.46%	34.89%	23.57%
15	53.76%	24.83%	28.94%
16	52.25%	27.07%	25.18%
17	37.45%	30.14%	7.31%
18	35.80%	18.99%	16.81%
19	60.16%	32.08%	28.08%
20	41.35%	20.37%	20.99%
Average Value	38.17%	20.09%	18.07%

Chapter 5

Conclusion and Future Work

Nowadays, cloud computing is becoming more and more popular everywhere. It saves a lot of money and avoids unnecessary spending for many individuals, companies and organizations. The number of large-scale data centers is growing fast in large cloud service companies like Amazon, Google, and so on. Unavoidably, server utilization and power consumption are two important aspects for the cloud computing industry. They are correlated to each other as resources with low utilization will consume a lot of energy.

In this thesis, the work proposed by Aazam et al in [3] is extended to improve the performance of a broker-based resource estimation model. Instead of using the same scheme for assigning resources, the proposed model uses different models depending on the situation. In other words, the new model can react to the changing environment of the cloud. Several simulations are designed and implemented for testing the model based on CloudSim and the simulation results show that the proposed model performs better than the model in [3], and it helps improve resource estimation, increases the resource utilization and hence reduces the unnecessary power consumption. From the simulation results, the performance of each model in the simulation meets the expectations from the mathematical analysis. In the real-time simulation discussed in the last chapter, the resource utilization of the combined model is improved by 18%.

The rest of this chapter is organized as follow. First, we come back to the research objectives outlined in Chapter 1 and go over the major contributions of this thesis. Finally, some ideas about how we can improve the research and extend the existing work will be presented.

5.1 Contributions, Results and Applications

In this thesis, we first started out with a literature review on the area of cloud computing. More specifically, inter-cloud computing and resource management in

cloud computing were studied to get some basic knowledge of this area. Based on our findings, the model in [3] stood out due to its innovative way of assigning resources. Starting with the model in [3], we defined three research objectives. These research objectives were all met and led to the following contributions:

- Four models were proposed implementing different strategies for aligning resources based on the resource estimation model in [3]. The factors used to do resource estimation in the proposed model are different than the factors used in the old resource estimation model. For example, the price of the service is not included for resource estimation while the actual amount of requested resources is included. For each module, the pros and cons were analyzed and supported mathematically and the performances of all modules were compared under several criteria such as server utilization, the amount of allocated resources, and customer experience.

- A decision tree was created containing six different situations considering different parameters such as current server utilization, the loyalty of the customer, price of the service and so on. Each situation is analyzed and matched to the most suitable module to achieve the most reasonable resource estimation according to their features. By allocating different modules to different scenarios, higher resource utilization and more reasonable resource allocation could be achieved. Then a new reactive prediction resource estimation model is proposed based on the decision tree. It consists of the four modules proposed before which deal with all the situations mentioned in the decision tree. The integrated model is called "reactive" because it can react to its environment dynamically.

- The simulation is based on CloudSim to test the performance of the proposed model and different modules under different situations. The differences between the modules are shown with simulation results which are similar to the mathematical analysis. Model 2 allocates more resources than other models do

while model 1 allocates the fewest resources in general. Model 3 and model 4 are in the middle but they focus on different kinds of customers as model 3 is suitable for requests from new customers or disloyal customers while model 4 is good for middle level and loyal customers.

- Real-time simulations were implemented to compare the performance of the new model and the old model. In the time period, there are different types of requests from different kinds of customers at random time stamps. From the simulation results, it is proved that the proposed reactive prediction resource estimation model performs better than the old resource estimation model as the overall average server utilization is higher when the broker implements the new resource estimation model, which reduces unnecessary waste of resources and power consumption.

According to the research objectives in Chapter 1, all the objectives are successfully achieved. The new reactive prediction model is designed and tested with simulations. Several parameters are considered for classifying different situations. Each module under this model is suitable for different situations and they are analyzed both mathematically and from the simulation result. Besides, a pledge system is added to the model so that disloyal customers are able to get the service they want by paying the pledge.

5.2 Future Work

There are still many things we can do to improve or extend this research. Below the possible improvements are listed:

- The modeling part could be improved using real-life data like Google Traces. This data could be analyzed to get the actual habits and behaviors of cloud customers. We can also find the busy time periods of one day or one week which could be utilized to develop different strategies. Then, based on the real data, some machine learning algorithms could be used to do data analysis on the user's

records and hence a more realistic model can be built to achieve better prediction and resource estimation.

- In this thesis, the four models could have better performance than the old model. We may evaluate if other models can provide better performance in those scenarios by adding more parameters.

- In this thesis, the pricing model uses a basic simple model. The customer pays according to the amount of allocated resources actually used. If customers relinquish the service, they will get reimbursed by the broker. But in the future, the pricing model could be set differently for different requests according to the loyalty of the customer, the history of the customer, current server utilization, etc. Moreover, the reimbursement strategy can also be enhanced by considering more parameters like quality degradation, the cost of reallocation of the resources and so on.

- In the thesis, the power consumption of the broker and cloud providers and the maintenance cost of different types of resources are not considered in the model. In the future, when we collect those parameters, it will improve the whole model because the cost and income of both provider and customer side will be more realistic. More specifically, each module under the proposed model can be optimized by building in some variables for income and expenses using those parameters. Finally, resource allocation decisions could be made based on an optimization model considering server utilization, costs, type of customers, etc.

- In the simulation, it is assumed that all requests are asking for the same amount of CPU, memory and RAM so that it is easier to calculate the server utilization. In the future, it could be improved to find another strategy to calculate the server utilization to make the simulation more realistic.

List of References

[1] J. Whitney, P. Delforge, “Data Center Efficiency Assessment”, Proceedings of Natural Resources Defense Council, 2014.

[2] I.S. Moreno, P. Garraghan, P. Townend, J. Xu, "An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models," Proceedings of Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on , pp.49-60, 25-28 March 2013.

[3] M. Aazam, E.N. Huh, “Broker as a Service (BaaS) Pricing and Resource Estimation Model”, Proceedings of 2014 IEEE 6th International Conference on Cloud Computing Technology and Science, 978-1-4799-4093-6/14 © 2014 IEEE DOI 10.1109/CloudCom.2014.57.

[4] Q. Zhang, L. Cheng, R. Boutaba, “Cloud computing: state-of-the-art and research challenges”, Proceedings of Internet Services Applications (2010): 7–18 DOI 10.1007/s13174-010-0007-6, © The Brazilian Computer Society 2010.

[5] XenSource Inc, Xen, available at: www.xensource.com. Last visit: Feb. 2016.

[6] Kernal Based Virtual Machine, available at: www.linux-kvm.org/page/ MainPage. Last visit: Feb. 2016.

[7] VMWare ESX Server, available at: www.vmware.com/products/esx. Last visit: Feb. 2016.

[8] A.M. Tripathi, R. Beg, “Cloud Computing – Architecture, Applications and Advantages” Proceedings of International Computer Technology and Applications, Vol 4(1), pp. 36-42, Jan-Feb 2013.

[9] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf, “NIST Cloud Computing Reference Architecture”. Proceedings of National Institute of Standards and Technology Special Publication 500-292, Sept. 2011.

[10] Amazon Elastic Computing Cloud, available at: aws.amazon.com/ec2. Last visit: Feb. 2016.

[11] S. Ghemawat, H. Gobioff, S.T. Leung, “The Google File System”, Google, DOI: 10.1145/1165389.945450, ISBN: 1581137575, ISSN: 01635980.

[12] Hadoop Distributed File System Architecture Guide, available at: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Last visit: Feb. 2016.

[13] W.C. Shih, S.S. Tseng, C.T. Yang, “Performance Study of Parallel Programming on Cloud Computing Environments Using MapReduce,” Proceedings of 2010 International Conference on Information Science and Applications, DOI: 10.1109/ICISA.2010.5480515, ISBN: 978-1-4244-5942-1.

[14] J. Dean, S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Proceedings of 6th Symposium on Operating Systems Design and Implementation, DOI: 10.1145/1327452.1327492, ISBN: 9781595936868, ISSN: 00010782.

[15] J. Lloret, M. Garcia, J. Tomas, J.P.C. Rodrigues, “Architecture and protocol for intercloud communication,” Proceedings of Information Sciences, Volume 258, 10 February 2014, Pages 434-451, ISSN 0020-0255, available at: http://dx.doi.org/10.1016/j.ins.2013.05.003.

[16] R. Buyya, R. Ranjan, R. N. Calheiros, “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit : Challenges and Opportunities.” Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, June 21 - 24, 2009.

[17] R. Buyya, R. Ranjan, R.N. Calheiros, “Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services”. Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing, 13-31, 2010.

[18] F. Jrad, J. Tao, A. Streit, “Simulation-based evaluation of an intercloud service broker,” Proceedings of Third Internet Conference of Cloud Computing and GRIDs Virtualization, pp. 140–145, 2012.

[19] M. Nir, A. Matrawy, M. St-Hilaire, “Optimizing Energy Consumption in Broker-Assisted Cyber Foraging Systems,” Proceedings of Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on, pp.576-583, 13-16 May 2014.

[20] M. Nir, A. Matrawy, M. St-Hilaire, “An energy optimizing scheduler for mobile cloud computing environments,” Proceedings of Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference, pp.404-409, April 27 2014-May 2 2014.

[21] Y.C., A. Zomaya, “Energy efficient utilization of resources in cloud computing systems,” Proceedings of The Journal of Supercomputing, Springer Science Business Media, 2010, pp. 268-280.

[22] Y. Yuan, W. Liu, “Efficient resource management for cloud computing,”

Proceedings of System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2011 International Conference on , vol.2, pp.233-236, 22-23 Oct. 2011.

[23] Google App Engine: Platform as a Service, available at: https://cloud.google.com/appengine/docs. Last visit: Feb. 2016.

[24] Amazon EC2, available at: https://aws.amazon.com/ec2/. Last visit: Feb. 2016.

[25] Y. Demchenko, M.X. Makkes, R. Strijkers, C. Laat, "Intercloud Architecture for interoperability and integration", Proceedings of Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on , pp.666-674, 3-6 Dec. 2012.

[26] S. Sotiriadis, N. Bessis, N. Antonpoulos, "Decentralized meta-brokers for inter-cloud: Modeling brokering coordinators for interoperable resource management," Proceedings of Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference, pp.2462-2468, 29-31 May 2012.

[27] H. Li, C. Wu, Z. Li, F.C.M. Lau, "Profit-maximizing virtual machine trading in a federation of selfish clouds," Proceedings of INFOCOM, 2013 Proceedings IEEE , pp.25-29, 14-19 April 2013.

[28] A. Leivadeas, C. Papagianni, S. Papavassiliou, "Efficient Resource Mapping Framework over Networked Clouds via Iterated Local Search-Based Request Partitioning," Proceedings of Parallel and Distributed Systems, IEEE Transactions on , vol.24, no.6, pp.1077-1086, June 2013.

[29] M. Chowdhury, M.R. Rahman, R. Boutaba, "ViNEYard: Virtual Network

Embedding Algorithms with Coordinated Node and Link Mapping," Proceedings of IEEE/ACM Trans. Networking, vol. 20, no. 1, pp. 206-219, Feb. 2012, doi: 10.1109/TNET.2011. 2159308.

[30] D. Dietrich, A. Rizk, P. Papadimitriou, "Multi-Provider Virtual Network Embedding With Limited Information Disclosure," Proceedings of Network and Service Management, IEEE Transactions on , vol.12, no.2, pp.188-201, June 2015.

[31] W. Wang, D. Niu, B. Li, B. Liang, "Dynamic Cloud Resource Reservation via Cloud Brokerage", Proceedings of 2013 IEEE 33rd International Conference on Distributed Computing Systems, 1063-6927/13 © 2013 IEEE, DOI 10.1109/ICDCS.2013.20.

[32] R. Miller, "Google's Energy Story: High Efficiency, Huge Scale." Available: http://www.datacenterknowledge.com/archives/2011/09/08/googles-energy-story-high-efficiency-huge-scale.

[33] A. Beloglazov, R. Buyya, "Energy Efficient Resource Management in Virtualized Cloud Data Centers," Proceedings of Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on , pp.826-831, 17-20 May 2010.

[34] Z. Gao, "The Allocation of Cloud Computing Resource Based on The Improved Ant colony Algorithm", Proceedings of 2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics, 978-1-4799-4955-7/14 © 2014 IEEE DOI 10.1109/IHMSC.2014.182.

[35] A. Nathani, S. Chaudhary, G. Somani, "Policy based resource allocation in IaaS cloud", Proceedings of Future Generation Computer Systems: 28(2012) 94-103, DOI:10.1016/j.future.2011.05.016.

[36] Y. Shi, X. Jiang, K. Ye, “An energy-efficient scheme for cloud resource provisioning based on CloudSim”, Proceedings of 2011 IEEE International Conference on Cluster Computing, 978-0-7695-4516-5/11 © 2011 IEEE, DOI: 10.1109/CLUSTER.2011.63.

[37] W. Lee, H. Teng, R. Hwang, “Optimization of Cloud Resource Subscription Policy”, Proceedings of 2012 IEEE 4th International Conference on Cloud Computing Technology and Science Optimization, 978-1-4673-4510-1/12©2012 IEEE.

[38] C. Wang, C. Yang, “A Prediction Based Energy Conserving Resources Allocation Scheme for Cloud Computing”, Proceedings of 2014 IEEE International Conference on Granular Computing, 978-1-4799-5464-3/14 ©2014 IEEE.

[39] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, “Energy-efficient cloud resource management,” Proceedings of Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference, pp.386-391, April 27 2014-May 2 2014.

[40] A. Biswas, S. Majumdar, B. Nandy, A. El-Haraki, “Automatic Resource Provisioning: A Machine Learning Based Proactive Approach,” Proceedings of Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference, pp.168-173, 15-18 Dec. 2014.

[41] I.S. Moreno, P. Garraghan, P. Townend, J. Xu, “An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models,” Proceedings of Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on , pp.49-60, 25-28 March 2013.

[42] M. Aazam, E.N. Huh, “Advance Resource Reservation and QoS Based Refunding in Cloud Federation”, Proceedings of Globecom 2014 Workshop - Cloud Computing Systems, Networks, and Applications, 978-1-4799-7470-2/14 ©2014 IEEE.

[43] “How AWS Pricing Works,” vol. 2015, no. June, pp. 1–15, Amazon Whitepaper, 2015.

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