Banking Theory Based Distributed Resource Management and Scheduling for Hybrid Cloud Computing

Hao Li

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Computing and Engineering
University of Huddersfield
July 2013

Table of Contents

Abstract

Cloud computing is a computing model in which the network offers a dynamically scalable service based on virtualized resources. The resources in the cloud environment are heterogeneous and geographically distributed. The user does not need to know how to manage those who support the cloud computing infrastructure. From the view of cloud computing, all hardware, software and networks are resources. All of the resources are dynamically scalable on demand. It can offer a complete service for the user even when these service resources are geographically distributed. The user pays for only what they use (pay-per-use). Meanwhile, the transaction environment will decide how to manage resource usage and cost, because all of the transactions have to follow the rule of the market. How to manage and schedule resources effectively becomes a very important part of cloud computing, and how to setup a new framework to offer a reliable, safe and executable service are very important issues.

The approach herein is a new contribution to cloud computing. It not only proposes a hybrid cloud computing model based on banking theory to manage transactions among all participants in the hybrid cloud computing environment, but also proposes a "Cloud Bank" framework to support all the related issues. There are some of technology and theory been used to offer contributions as below:

1. This thesis presents an Optimal Deposit-loan Ratio Theory to adjust the pricing between the resource provider and resource consumer to realize both benefit maximization and cloud service optimization for all participants.
2. It also offers a new pricing schema using Centralized Synchronous Algorithm and Distributed Price Adjustment Algorithm to control all lifecycles and dynamically price all resources.
3. Normally, commercial banks apply four factors mitigation and to predict the risk: Probability of Default, Loss Given Default, Exposure at Default and Maturity. This thesis applies Probability of Default model of credit risk to forecast the safety supply of the

resource. The Logistic Regression Model been used to control some factors in resource allocation. At the same time, the thesis uses Multivariate Statistical analysis to predict risk.

4. The Cloud Bank model applies an improved Pareto Optimality Algorithm to build its own scheduling system.
5. In order to archive the above purpose, this thesis proposes a new QoS-based SLA-CBSAL to describe all the physical resource and the processing of thread.

In order to support all the related algorithms and theories, the thesis uses the CloudSim simulation tools give a test result to support some of the Cloud Bank management strategies and algorithms. The experiment shows us that the Cloud Bank Model is a new possible solution for hybrid cloud computing.

For future research direction, the author will focus on building real hybrid cloud computing and simulate actual user behaviour in a real environment, and continue to modify and improve the feasibility and effectiveness of the project. For the risk mitigation and prediction, the risks can be divided into the four categories: credit risk, liquidity risk, operational risk, and other risks. Although this thesis uses credit risk and liquidity risk research, in a real trading environment operational risks and other risks exist. Only through improvements to the designation of all risk types of analysis and strategy can our Cloud Bank be considered relatively complete.

Key words: ***Hybrid Cloud Computing, Bank Model, Microeconomics, Scheduling, Risk Mitigation, Pricing, CB-SLA***

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text, and a list of references is given**.**

Hao Li

July 27, 2013

Acknowledgements

I would like to thank my first supervisor, Professor Joan Lu for her supervision. I would also like to acknowledge my China supervisor, Prof. Shaowen Yao, for advising me in my research and supporting my career.

I also want to thank the School of Computing and Engineering at the University of Huddersfield for providing me with the opportunity to study and for providing assistance throughout my research.

I want to thank my parents, Shouming Li and Mingxiao Wang who more than anyone else never gave up in supporting my studies. I also want to thank my wife, Yun Zhang, for her patience and for always staying by me during my studies and my son Xinyan Li who is always support me to do what I want. I would like to thank all of my friends in the School of Software, Yunnan University, China, for their kind words and encouragement.

Table of Contents

Table of Figures

List of Tables

List of Abbreviations

ASP Application Service Provider

AWS Amazon Web Service

CA Consumer Agency

CB Cloud Bank

CB-SLA Cloud Bank Service Level Agreement

CC Cluster Controller

CCA Cloud Consumer Agency

CLC Cloud Controller

CRACloud Resource Agency

CRP Cloud Resource Provider

CRS Cloud Resource Consumer

CRM Cloud Resource Manager

CLC Cloud Controller

CIS Cloud Information Server

CSA Cloud Service Agency

CSP Cloud Service Providers

DaaS Data as a Service

DBC Deadline and Budget Constrained

DR Default Ratio

EAI Enterprise Application Integration

EC2 Elastic Compute Cloud

GFS Google File System

HDFS Hadoop Distributed File System

IDC Internet Data Center

MP Market Price

MRS Marginal Rate of substitution

NC Node Controller

OGSA Open Grid Services Architecture

PA Provider Agency

PaaS	Platform as a Service
QoS	Quality of Service
QoR	Quality of Resource
RC	Relative Costs
RCC	Resource Classify Centre
RPC	Remote Procedure Call
RM	Resource Marker
RIM	Risk Manager
RR	Renting Recorder
RC-SLA	Resource Consumer-SLA
RP-SLA	Resource Provider-SLA
VP	Virtual Pool
VO	Virtual Organization
VM	Virtual Machine
VI	Virtual Infrastructure
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service Level Objectives
S3	Simples Storage Service
SC	Storage Controller
SC	Storage Controller
SSP	Storage Service Provider
SLM	Service level Management
SOA	Service Oriented Architecture
XM L	Extensible Markup Language
WSDL	Web Services Description Language
WSRF	OASIS Web Services Resource Framework
UDDI	Universal Description Discovery and Integration
UEC	Ubuntu Enterprise Cloud

Chapter 1Cloud Computing Resource Management and Scheduling Based on a Banking Model

1.1 Introduction

As part of the new technological revolution, cloud computing will fundamentally alter the way we do commerce. Indeed, nearly all IT resources may be provided as a cloud service: the application procedure, the computing power, the storage capacity, the networking, computing abilities, the communication services and even the development tools. The project began when Google and Amazon launched cloud computing in order to extend their infrastructure and provide services to their users. The virtualized resources composed the cloud, which provides services for the Internet user through the public interface, and bills the user according to the amount of service consumption. The user can use any available terminal devices to access the service at anytime and anyplace, which the cloud service provider provides.

The key feature of cloud computing is the ability to provide internet-based applications which are reliable, secure, fault-tolerant, and sustainable, and which have scalable infrastructure or services. These applications have different compositions, configurations and deployment requirements, and as a result developing scheduling and allocation strategies and quantifying them are very challenging problems for the various applications and modes of service in the cloud infrastructure. Several heterogeneous resources been used in actual cloud computing applications. Certain resources and functions require more cooperation in the cloud, which complicates resource management and scheduling. However, the present implemented network foundation cannot satisfy the requirements of the QoS. When service cooperation uses too many resources simultaneously, performance and efficiency decrease.

When the server provides the resources necessary for applications, then issue of service provision transforms alongside the issues of the resource management and assignment. An emerging science is only viable when it enters the commercial arena. With rapid development of computer science, the top tier service providers have gotten deeply involved with the lower level resource scheduling and management. Service is provided under the market environment, which means that the lower level resources management and scheduling are also influenced by the same market rules.

Cloud computing is committed to providing services using the pay-as-you-go model. Therefore, cloud computing infrastructure must be able to support various transactions between customers and suppliers of cloud agency platforms. Economic theory presents one of the most efficient resource deployment methods. It solves distributing resources to many autonomous using a simple, effective mechanism, and may also present an optimal solution or an approximate optimal solution to this problem. Dr. Buyya pointed out in an article[1]that rationality underlies the applied economics principles for grid resource management. He thinks that resource management models based on economic theory provide a more suitable distribution and complex environment.

1.2 Inspiration for Computational Clouds

After studying economic theory and the characteristics of cloud computing, we designed a mechanism to manage and schedule cloud resources based on the banking industry’s market theory. It mainly focuses on how cloud computing follows the real bank model to build a new cloud computing architecture. The purpose of this research is to deliver a new Cloud Bank Model which can offer new insight into developing public-based hybrid cloud infrastructure in the future. Inspired by deposit-loan transactions in commercial bank, we advance Cloud Banks[2] as shown in figure 1. In this model, the whole cloud computing environment is classified into three separate roles: cloud resource providers (CRP), Cloud Bank (CB), cloud resource consumers (CRS). The resource provider is equivalent to the depositors of commercial banks, corresponding resource consumers are similar to the borrower, and the Cloud Bank plays the role of a commercial bank. Cloud Bank pays the depositors a certain interest rate based on storage time, and charges the borrower according to the interest rate and the time over which tasks were executed. In order to balance supply, demand, and maximum profit for the Cloud Bank operators, this thesis will introduce the optimal deposit-loan ratio theory of commercial banks to the daily management of Cloud Banks.

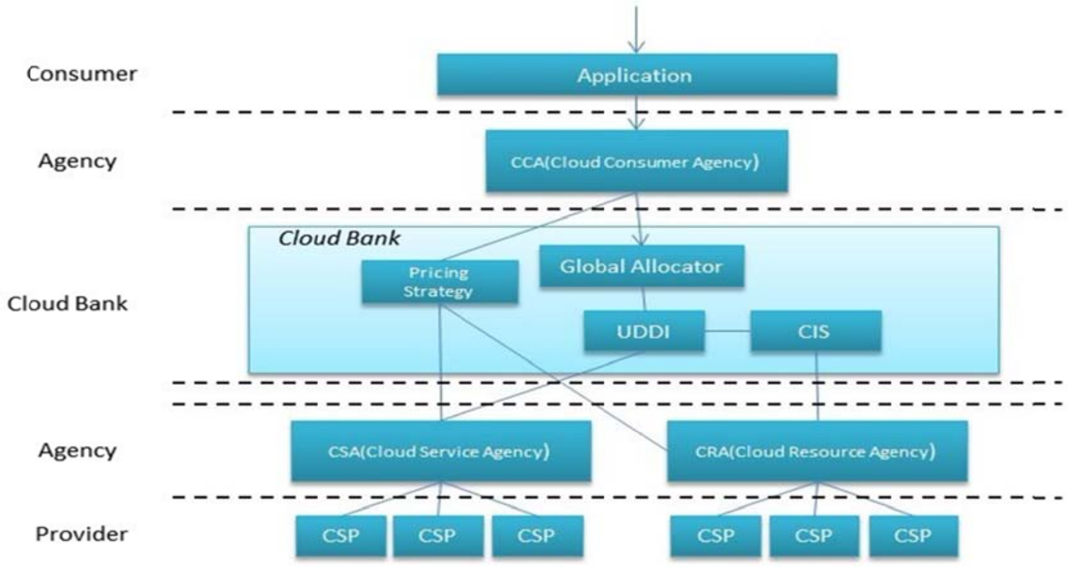


Figure 1. 1Architecture of the Cloud Bank model

The deposit-loan ratio controls the deposit interest rate and the lending rate. This regulation will realize a win-win situation for both the provider and the consumer while simultaneously maximizing the benefit of the Cloud Bank. This thesis will detail the characteristics and quantitative methods of cloud resources in detail using the price operation mechanisms of real banks.

1.3 Banking Model Based Cloud Computing Resource Management and Scheduling

The Cloud Bank model contains three levels: an application layer, a virtual organization (VO) layer, and a physical resources/facilities layer. These are further divided according to the three participatory roles in cloud computation: cloud resources consumer, cloud resources provider as well as the Cloud Bank agent. A QoS-based Cloud Banking model is introduced to make use of this hierarchical idea. This model also divides the QoS of cloud service into four parts according to their properties. There are logic resources QoS, system QoS, security QoS, and trust QoS in each level. The model uses the price release lever to control transactions inside and outside of the Cloud Bank, and used the quantification cloud currency as a cloud resource parameter, and considers the supply and demand situation of the market cloud resource.

The resource provider agent represents the economic advantage of the entire physical resources layer. The consumer agent represents the economic advantage of the all the consumers who use the cloud computing resource to achieve individual purposes. Since the hybrid cloud computer resource agent plays a bank role in the transaction, the service agents in the VO level provide the resource to the consumer and at the same time act as a buyer to the resource providers.

The transactions of those agents are based on the Service Level Agreement (SLA)[3]. All the agents use the SLA to display their requirement and characteristic parameters. In the bank model, the utility functions represent QoS requirements for each dimension. According to the different requirements of each job, it is possible to divide them into the different QoS dimensions. Once the client has the service requirement, the corresponding consumer's agent will be created automatically. The resource distributor will start to analyze the resource requirement in the VO. The resource distributor will get an amount of budget funds to finish the corresponding job. That budget funds amount is the same amount that the consumer must pay once their job is successful completed. Different consumer agents have different schema to distribute the budget funds; deadline, budget, the size of the data block and CPU running time could all represent the parameters of the job.

1.3.1 The Cloud Computing Requirements Analysis

The parallel computing and the grid computation have provided the most basic ideals for the cloud computation's development, which is the cloud computation development's initial stage. Actually, it is the commercial implementation of the concept of those computer sciences.

There are five main reasons for making cloud computing increasingly common, gradually developing into a mainstream computing model that would replace centralized mainframe computers and traditional distributed computing.

- Cloud computing has a good performance and performs even better than a centralized system. Also, it does not cost as much to get access to high-performance computing.
- Most of the applications are distributed. For example, industrial enterprise application management and sites are not in the same location.
- High reliability-- redundancy is not only a necessary condition for biological evolution, but also for information technology. Modern distributed systems have a

high level of fault-tolerant mechanisms to control all kinds of systems to achieve high reliability.

- Scalability-- buy a higher-performance computer or same performance computer for more than the cost of adding several PCs.
- Highly flexible-- compatible with different hardware vendors, compatible with low configuration machines and peripherals while achieving high performance computing ability[4].

1.3.2 The Technical Cloud Environment Requirement

The basic infrastructure technical cloud computing platform in the lab environment is built as follows:

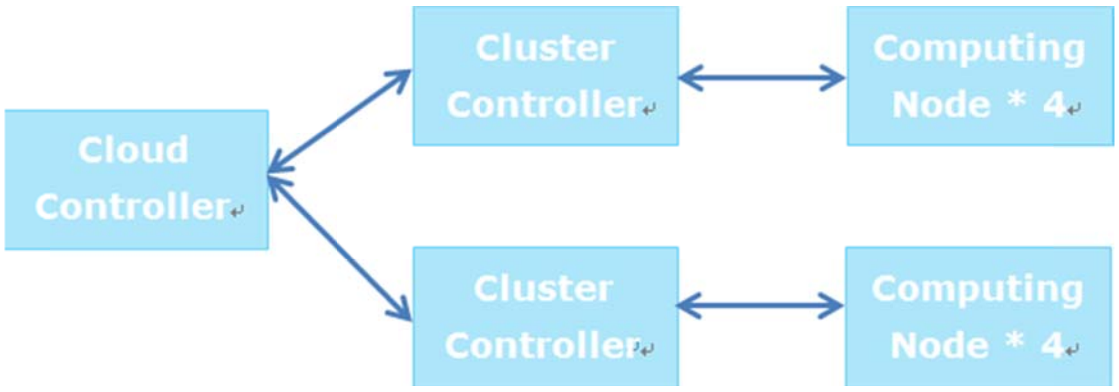


Figure 1. 2Topological Structure of Cloud Computing Platform

- Install Eucalyptus [5] software package as the cloud platform based on the Ubuntu [6] server in 11 personal computers. All of the PC's CPU must support virtualization technology.
- 3 PCs act as the "Front-end," where one acts as cloud controller and two of them as cluster controllers. The remainder of the 8 PCs will be divided into 2 groups, each group consisting of 4 PCs to act as the "computing-node".
- 1000M LAN switches connected 11 PCs.
- Make to run on the Ubuntu server. The image is a SaaS based software project, which includes a web container (Apache Tomcat), database server (MySQL) and web service based applications.
- Making load balance image. Installing HAProxy and using MySQL Cluster to apply database load balance.

- Through access instance of load balance, dynamic scheduling and application mirror to realize dynamic cloud computing services.

1.4 Contributions to Research

Our model is a novel and innovative cloud computing framework; it is first framework to use banking models and economic theory to manage and schedule cloud computing resources. Compared with the traditional cloud resource management systems, the new framework will deliver great value to all parties involved in the cloud computing, because the new framework is dynamicity with a simple processing mode to facilitate resource sharing. Moreover, this model presents a great opportunity to move cloud computing beyond academic research and into the commercial sector. The main contributions of new framework are listed below.

1.4.1 Banking Model for Cloud Computing Resource Management

The Cloud Bank model is a kind of hybrid cloud computing model based on the principles of microeconomics. In terms of resource organization, this thesis tries to absorb distributed and dynamic resources into a "resource pool" bank, and adopts the "deposit-loan" pattern, the architecture of which is similar to real commercial banks. There are three basic roles in the Cloud Bank model: the Cloud Resource Provider (CRP), the Cloud Bank (CB) and the Cloud Resource Consumer (CRC). The logical structure is shown in Fig 1.3:

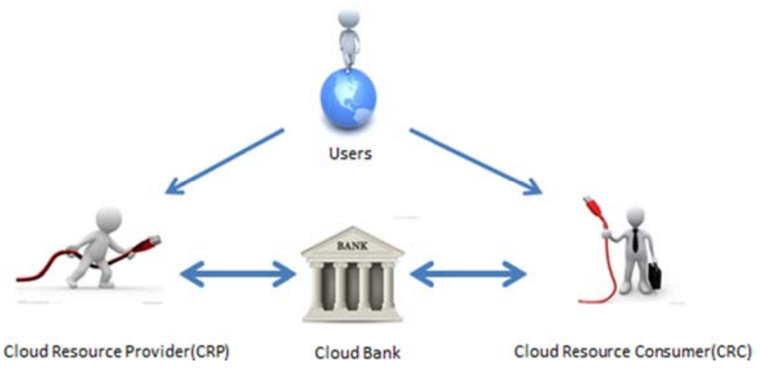


Figure 1. 3 Architecture of the Cloud Bank model

1.4.2 Optimal Deposit-loan Ratio

The deposit-loan ratio of the Cloud Bank refers to the proportion of total loans to total deposits. As mentioned above, this thesis compares resources stored in the Cloud Bank to bank deposits, and resources borrowed to loans. The thesis assumes that the more we lend stored resources the better. However, economic research points that transforming deposits into loans is not necessarily the more the better, because there exist certain commercial risks. Because the bank has to pay for the depositor, once the bank cannot loan the money to the loaner, the bank will lose money. The most serious condition can lead banks to declare bankruptcy in which withdrawals exceed loaned resources before the loaned resources are returned upon task completion. In the cloud environment, these situations are frequent happen and uncontrolled. When the cloud bank lacks for the resource to loaner, the new resource must fill the resource pool. The Cloud Bank must reserve spare resources to ensure the task finished on time, for otherwise the Cloud Bank will suffer loan default loss. Therefore, the deposit to loans conversion ratio is not the more the better, and there exists an optimal deposit-loan ratio. Only under the premise of this ratio can the Cloud Bank maximize benefits for all of participants. Therefore, as indicated by this research, this thesis expects to find the right deposit-loan ratio in the Cloud Bank management model. The Cloud Bank adjusts deposit and loan interest rates through this ratio to realize maximum benefit for all participants and optimize the cloud services.

In the cloud computing environment, resources providers know the resource volume provided by others, so if they choose as many as they want resources to achieve maximize their own interests. Cloud resource providers offer the resources with the goal of obtaining maximum usage. The relationship among them is a transaction, and this transaction’s final result is that participants using or lending the various resources maximize their own interests, and at the same time the resources are optimally used.

1.4.3 Risk Mitigation and Prediction Management

The resources in the Cloud Bank model are managed based on the economic principles. Developed from the commercial bank model, the Cloud Bank has some uncertainties that are similar to the commercial banks. For example, the depositors in the commercial bank may withdraw their money at any time, which is very similar to the situation in which the resources in the Cloud Bank may be unavailable at any time.

The environment of cloud computing is dynamic and distributed. The resource providers are distributed physically and the provided resources heterogeneously, resulting in many uncertainties that could affect the reliability of the Cloud Bank. For example, the resources could be unavailable because of broken cables. These uncertainties could lead to risks that in turn could seriously affect the reliability and reputation of the Cloud Bank. Thus, the risk mitigation and management is a very important part of the whole Cloud Bank model.

For various reasons, the risks could be divided into six types: systematic or market risk, credit risk, counterparty risk, liquidity risk, operational risk, and legal risk. The risks are not all borne by the bank itself. Several methods are developed in this thesis to mitigate risk mitigation and predict risk.

1.4.4 Pricing Scheme of the Cloud Resource over the Lifecycle

In order to describe the features of Cloud Bank model, the thesis presents the lifecycle of Cloud Bank. The two stages in the lifecycle are shown in Fig1.4:

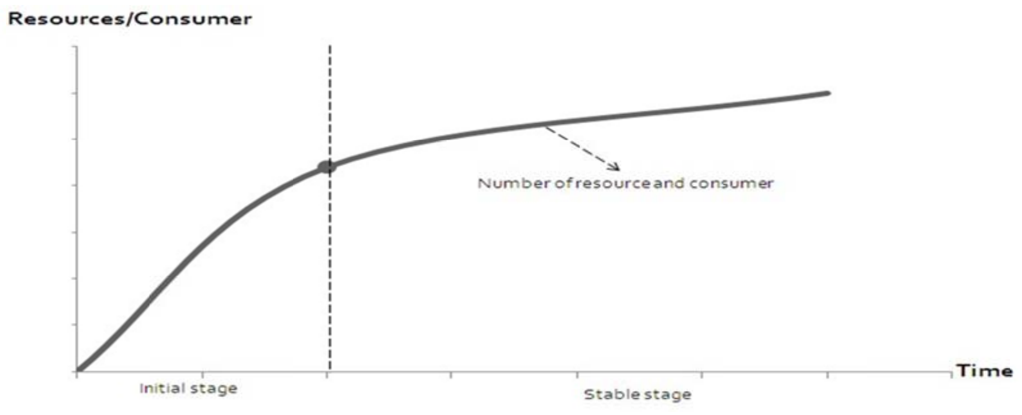


Figure 1. 4 The lifecycle of the Cloud Bank

In the initial stage, there are few resources in the Cloud Bank and few consumers have joined. The number of consumers increases with time. The resource providers operate a monopoly on provision, for initially the price of resources is decided solely by interaction among resource providers. The price strategies among the different CRPs are the same. Two common monopoly models are: Cournot model[7] and the Bertrand model[8]. Cournot model guesses the price under production. The Bertrand model uses price as a strategy to infer monopoly pricing. The solution of Cournot equilibrium is the Nash equilibrium. The Nash equilibrium is when participants maximize their own interests while the interests of others are also maximized. Therefore, maximizing the interests of various participants is possible the cloud computing environment using the Cournot equilibrium theory.

When the curve reaches a certain point, the bank enters into a balanced status, and the second stage begins.

The second stage is the stable stage, with increasing numbers of consumers, and random producers beginning to inject resources into the resource pool. Finally, the Cloud Bank will reach a stable state. This thesis divides the resource pricing strategy into two stages as shown in Figure 1.5.

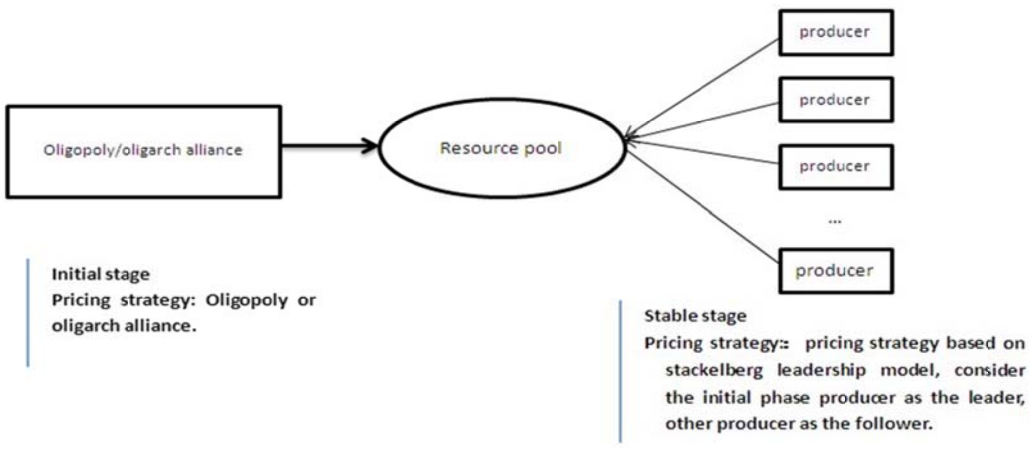

producer
producer
producer
...
producer
Oligopoly/oligarch alliance
Resource pool
Initial stage
Pricing strategy: Oligopoly or oligarch alliance.
Stable stage
Pricing strategy:: pricing strategy based on stackelberg leadership model, consider the initial phase producer as the leader, other producer as the follower.

Figure 1. 5 Resource pricing strategy

In the first stage, the resources in resource pool come from the Cloud Bank or oligarchs, so the pricing strategy can be called an oligopoly or another pricing strategy such as the pricing strategy based on the Cournot equilibrium[9]. When the transaction state enters the stable stage, the resource price will fluctuate as random producers join and exit, so we use Stackelberg leadership model to guide the pricing strategy.

1.4.5 Cloud Computing Resource Management and Distribution Based on the Pareto Equilibrium

With scientific and technological development, a number of subjects began to cross-apply actual projects. The introduction of economic theory to computer science is a famous example of this trend. The Pareto optimal state is when a group of people obtain more profits without decreasing the profits of another group of people. Pareto optimal state is generally considered to be an optimal allocation of social resources. Termed Pareto improvement, it suggests that in some economic circumstances, people can improve their welfare or satisfaction through proper mechanisms without reducing any other person's satisfaction or benefits.

Welfare economics primarily discusses the Pareto optimal state[10]from the view of customers and production. The situation of the Pareto optimal equilibrium and the management of the resource customers and resource providers in the cloud computing are a perfect match. In the Cloud Bank, distributing computer resources effectively among tasks from cloud costumers is a crucial problem. Hence, research can establish a mechanism applying Pareto optimal theory to allocate computing resources in cloud computing. After all, resource distribution in social and economic fields is more complicated and has been researched more thoroughly. This research can apply these theories easily and smoothly to computer science by changing only several factors.

The traditional Pareto theory is a two-dimensional resource distribution strategy[11]. There are two resource providers and two consumers. However, in the real cloud computing environment there are n×m dimensions. In this thesis, the author expands the two relations to fulfil the n×m condition and in order to offer a resource distribution strategy based on the bank model.

1.4.6 Building a Real Cloud Computing Platform Based on Open Sourcing

In order to test the Cloud Bank theory, we set up a real cloud computing platform based on open sourcing. The most important part is the hypervisor of cloud computing, the role of which is shown in figure 1.6

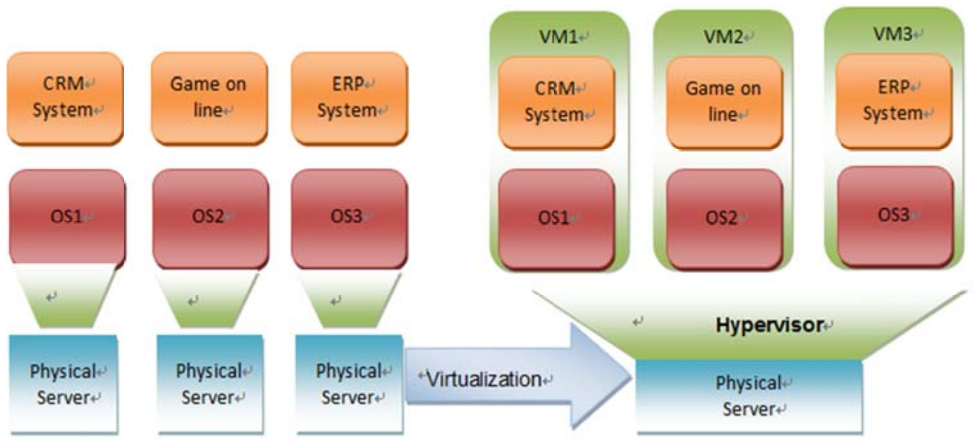


Figure 1. 6 the Role of the Hypervisor in the Cloud Computing

This thesis developed two open source based cloud computing test beds, Eucalyptus and OpenStack, and give a test result to support the IaaS based Cloud computing model. Basically, the researchers needed a hypervisor to support the different operating systems and the VM. It makes all the computing nodes into naked nodes capable of joining any necessary computing process, eliminating concerns about operating systems or any other environment.

1.5 Organization

The rest of this thesis is organized as follows.

Chapter 2 presents background research about cloud computing and resource management and scheduling. It discuss the evolution of cloud computing technology, while simultaneously giving a study about cloud computing. It also discusses the different types of cloud computing. Finally, it provides a survey about cloud resource management.

Chapter 3 provides background research about computational economy theory. This paper tries to use a metaphor for effective management of distributed resources and application scheduling. It discusses the principle of how to use real-world economic models and strategies. It shows the relationship between economic theory and resource management of cloud computing.

Chapter 4 pertains to problem identification, and presents four key issues for hybrid cloud computing: resource accounting, resource scheduling, resource security and QoS issue. This section will clearly describe these problems.

Chapter 5 presents the architecture and implementation of a Cloud Bank resource broker that uses a computational economics driven framework for managing resources and scheduling applications. This chapter gives a general overview of the Cloud Bank model. It discusses how cloud computing follows the real bank to do transaction, ascertains optimal deposit-loan ratio and the pricing schema for the cloud computing, and discusses how the Cloud Bank manages risk mitigation and predicted risk in the transactions.

Chapter 6 discusses the design and implementation of the Cloud Bank by following the risk-mitigation model in real banks to the cloud computing environment and analysing the types of risk and their features. As previously stated, the cloud computing environment is far from homogenous, with diverse provisions and providers scattered geographically, all of which contributes to risk. Moreover, the Cloud Bank model is a resource management tool based on economic principles. Developed from the commercial bank model, the Cloud Bank has some uncertainties that are similar to the commercial banks. Thus, this chapter concludes by setting up a risk mitigation model and providing the corresponding algorithms in order to address this issue.

Chapter 7 presents the pricing of cloud computing resources as a very important issue for successful transactions. In this chapter, the Cournot Equilibrium is introduced to address the initialization of the resource price. The Cloud Bank needs a way to price all the newly entered resources according to the individual situation of the Cloud Bank. The Stackelberg model leads the price to the optimization state over the course of operation. All related models and algorithms are also introduced in this chapter.

Chapter 8 demonstrated how to use the Pareto optimality theory to give a concrete definite resource distribution strategy and algorithm. This chapter shows that different economic models offer different services and consumption prices. It is the business strategy level for the Cloud Bank. The cloud computing multi-resources market model is a wholly competition market. According to the economic general equilibrium theory, the resource distribution that satisfies the equilibrium price is the most reasonable and most effective. The multi-commodity market model provides superior resource deployment, the ideal situation, and a simple economic model. All related models and algorithms are introduced in this chapter. At the same time, the thesis also presents improved min-min algorithms to demonstrate the effectiveness of the Cloud Bank model and related strategies. The conclusion provides the test results and analysis.

Chapter 9 presents the real cloud computing platform in the IaaS category. It offers the simulation environment in order to experiment with the Cloud Bank theory. IaaS can provide servers, operating systems, disk storage, and database/information resources. In this chapter, it is shown how the IaaS supports the experiment.

Chapter 10 presents a summary of the thesis, and delineates directions for future research in banking-based cloud resource management.

Chapter 2 Background Study on Cloud Computing and Resource Management and Scheduling

Resource scheduling strategies are integral to comprehensive sharing and collaboration with each other. Since the resources are owned by different organizations, the cloud computing platform should permit these resources to join dynamically and leave freely. The cloud resource scheduler needs to shield this characteristic from users. Obviously, it is difficult to achieve unified resource management in clouds, and a static way to manage resources is inappropriate. From a strategic viewpoint, the free market economy is a simple and effective method for allocating resources among multiple self-interested individuals, and thus provides an approximate optimal solution.

In the free market, prices reflect changes in supply and demand along with the balance between supply and demand to obtain resource optimization. Such a dynamic and coordinated mechanism is appropriate for cloud resource characteristics. This thesis mainly focuses on aspects of resource scheduling strategy in the Cloud Bank model based on the existing cloud resource management paradigm. This model borrows from economic theory in order to maximize the benefits for all the parties.

2.1 Computational Clouds

Cloud computing is a development of distributed processing. The basic concept for the cloud commercial realization in computer science comes from parallel computing and grid computing.

The biggest difference between cloud computation and traditional computation is that all the computing resources are come from the "Cloud". Cloud computing emphasizes the whole cloud concept. However, the cloud computation and traditional computation are not essentially different.

Cloud computing has two typical characteristics. First is integration, as all the members of the cloud unite to form one entity. Members are connected together by virtue of the configuration of organization. The second is simplification. In the current hybrid cloud computing system, the member is only a simple service or resource provider/consumer, although this member can possibly be a sub-cloud computing system.

In order to fully understand cloud computing, it is important to first understand two key concepts: computation and algorithm. Starting from the 1930s, the work of Kurt Gödel, Alonzo Church and Turing lead to the establishment of a mathematics branch of computation: recursion theory and computability theory. This theory believed that from a known string start, according to certain rule change string, passes through the limited step, and finally obtains the string that satisfies stipulated in advance. This conversion process was called computation. In short, the computation is the continuous transformation of the string.

An algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning[12]. In other words, that is the computer problem solving process. In this process, regardless of being forms the problem solving mentality or the write program, both of them are implementing some algorithm. The former is the algorithm that the inference realizes and the latter is the algorithm that the operation realizes.

2.2 The Evolution of Cloud Computing Technology

How did cloud computation slowly develop from desktop machine systems to parallel computing, distributed computing, and grid computation? What do the above computation patterns have in common and what is different? After investigating these questions, the following part of this chapter will show how cloud computing has come to face so many challenges.

2.2.1 Parallel Computation

For a long time, people have used serial computing to process computation, but with the computing job getting bigger and bigger, the serial computation is unable to satisfactorily solve problems as required.

Traditionally, software has been written for serial computation:

- To be run on a single computer having a single Central Processing Unit (CPU);
- A problem is broken into a discrete series of instructions.
- Instructions are executed one after another.

- Only one instruction may be executed at any moment in time [13].

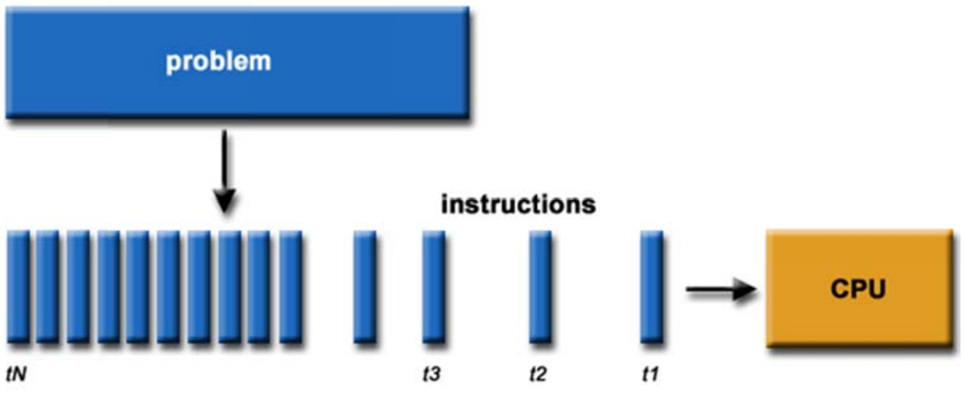


Figure 2. 1Logic Model of Serial Computing[14]

In the simplest sense, parallel computing is the simultaneous use of multiple computing resources to solve a computational problem:

- To be able to run using multiple CPUs
- A problem is broken into discrete parts that can be solved concurrently
- Each part is further broken down to a series of instructions
- Instructions from each part execute simultaneously on different CPUs

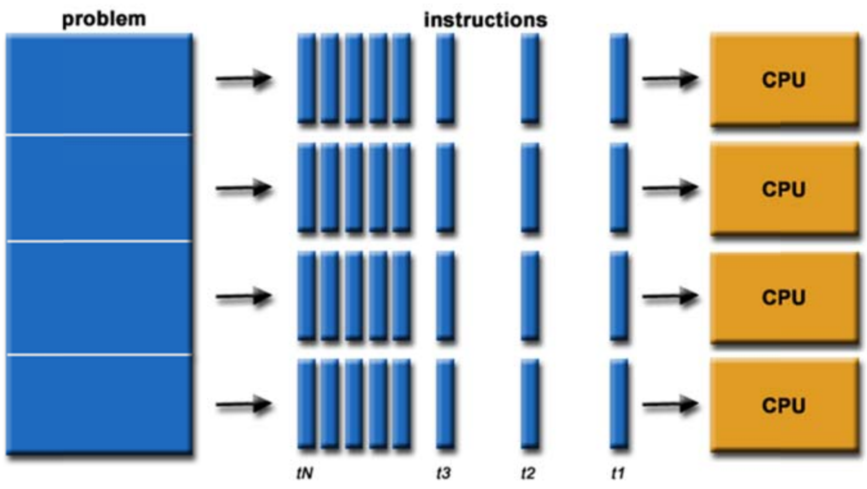


Figure 2. 2 Logic Model of Parallel Computing

There is one big problem with parallel computing. The jobs are divided into different whole task modules. These modules are then assigned into different computing nodes. If one of the modules been changed then the whole computing result will be affected.

2.2.2 Distributed Computing

Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs[16].

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one computer[17].

Distributed computing and parallel computing share the same basic principle. Both try to combine the capacity of numerous idle computers on the Internet together to solve some large-scale estimation problem. Both of them try to divide a large problem into many parts, and then assign the parts to many computers carry out the processing and finally synthesizing these results to obtain a result.

Distributed computing differs from parallel computing in that the task packages in distributed computing are independent, and some small fault in a task's computation cannot affect other task packages. Distributed computing submits identical tasks to multiple machines for execution. The results are then uploaded to the server to verify the accuracy and offer a conclusion.

2.2.3 The Main Difference Between Distributed Computing and Parallel Computing

The core concept behind both distributed computing and parallel computing is that big tasks are decomposed into small parts, and then assigned to many computers for the servers to process. In distributed computing, x small tasks are mutually independent. A computation fault in one task does not affect other tasks. Error correction can be achieved through executing the same task on different servers. However, in parallel computing, the task relations are very close. Computed results mutually influence each other. It is required that each task must be calculated absolutely correctly in synchronized time. This places a very high demand on computation accuracy and the synchronism request.

2.2.4 Grid Computing

Both parallel computing and distributed computing have provided the basic idea behind the development of grid computation, which is the initial stage of grid computation development.

2.2.4.1 The History of Grid Computing

In the 90’s, the global expansion of the Internet turned it into an effective tool that helps people to communicate and cooperate with each other. More importantly, tens of thousands of computing resources, data resources, software resource, all kinds of digital equipment and control systems constitute a resource pool on the Internet. At the same time, it also is an important carrier for the production and dissemination of knowledge. Thus, the incipience of the question of how multitudinous resources, linked physically, can provide joint services.

In 1992, Mr.Charlie and Larry Smarr of the US National Super Computation Application Center (NCSA) proposed the concept of “meta computation” in order to construct a virtual computer environment in a network[18].

In 1995, Tom De Fanti from the University of Illinois and Gong Rick from Argonne National Laboratory set up one of the earliest large meta-computing experiment according to the I-Way (Information Wide Area Year) plan[19]. The plan connected 17 computing centres through the nation's high-speed wide area network.

From the I-Way plan, Ian Foster from the Argonne National Laboratory and Carl Kesselman's team from the University of Southern California launched the Globus plans and developed the middleware for high performance distributed computing in 1996. In 1998, the concept for the blueprint of Grid Technology was introduced.

The Toolkit of Globus provides user authentication, grid resource allocation and management of the basic properties of the grid middleware. The software was provided as open source for the public, and was used by many research projects to build the grid computing. In fact, it has become a standard of the grid computing. However, Globus Toolkit just provides only basic components, but more comprehensive software is needed to build the grid computing framework.

In addition, Professor Miron Livny tried to freely use the distributed idle resources of the Internet. In 1985, he used the idle CPU of workstations in University of Wisconsin to process computations[20]. In 1991, Condor Program was successful with a concentration of 400 CPUs.

In 1997, Scott Kurowsk created the Entropia Company and successfully concentrated the idle personal computers with Internet access to achieve the same computing ability as the highest performing teraflop supercomputer. Meta computing and flexibility in using the idle CPU components are the essence of grid technology. Both of them are able to provide a single, virtual computer environment for distributed computers through the network.

In Japan, people can remotely operate computer. The global computing studies are in progress. From 1994, as a global computing study, the project concentrates on design and verifies remote procedure call (RPC). This protocol can allow a distributed server to perform the computing task of the remote client.

2.2.4.2 The Taxonomy of Grid Computing

According to the view of Ian Foster and Globus[21], the main areas of grid computing are distributed supercomputing, distributed instrumentation system, data intensive computing and tele-immersion.

Distributed supercomputing refers to connecting various supercomputers from different locations by high speed networks and has been agglutinated with grid middleware. It has more powerful computation ability than the single supercomputer. In fact, the original purpose of grid computing is to accommodate large-scale computing requirements. The typical application of grid computing in distributed supercomputing is SF Express[22]. It divided large Military Simulation into the distributed environment.

The second application is called the digital theory of relativity. It uses the grid to solve the Einstein relativistic equation and to simulate the law of motion of the heavenly body.

Distributed instrumentation system refers to distribution grid management systems offering an easy and friendly method to access the various valuable instruments and receive results remotely. Before the emergence of the grid technology, the hardware and software environment limited efforts to access network equipment or instrument data. Applications can only achieve some minimum requirements. The distributed instrumentation grid system became as a very easy and flexible management system.

Parallel computing is driven by compute-intensive applications, especially some of the grand challenges, which promote high-performance parallel architecture, the programming environment, large-scale visualization research and so on. Compared with computer-intensive applications, the application of data-intensive computing is more than compute-intensive applications. It corresponds to the data grid as it is more focused on data storage, transmission and processing. The computational grid is more focused on how to promote the ability of computing, so the focus and implementation techniques are different. For example, the DataGrid project from the European Atomic Energy Centre falls into the data grid application. A representative project in this area is the Xport[23] Project supported by the U.S. Department of Energy. Based on Globus, it provides remote instrument use such as project planning, instrument operation, data acquisition, filtering and analysis functions, which significantly simplifies the design of a giant molecular crystal structure and implementation. It is possible to visualize image of the internal structure of the crystal remotely.

Tele-immersion is a technology implemented alongside the Internet that enables users in different geographic locations to come together in a simulated environment to interact with each other. Users will feel like they are actually looking, talking, and meeting with each other face-to-face in the same room [24]. This term was first released by the University of Illinois at Chicago Electronic Visualization Laboratory EVL (Electronic Visualization Laboratory) in October 1996. It is a typical remote grid computing application that centralizes a shared virtual environment. It also can visually show the results of high-performance computing and database. The EVL in collaboration with various partners developed some remote immersive virtual reality applications, such as a virtual museum. Visitors entering the network can not only travel in the virtual city, but also share the virtual space and communications with other visitors. At the same time, EVL and the Interactive Computing Environment Laboratory CEL launched a major project, NICE（Narrative Immersive Constructionist / Collaborative Environments).

2.2.4.3 The Taxonomy of Grid Resource Management

1. Traditional scheduling algorithms

The traditional scheduling algorithms are Round Robin, Weight Round Robin, minimum connection scheduling and its improvement algorithm, Destination Hashing Scheduling, Source Hashing Scheduling and so on. All of them are very clear and tidy, but their performance is not good enough.

2. Heuristic intelligent scheduling algorithm

The grid task-scheduling problem has been proven to be NP hard, so most grid task scheduling algorithms is based mainly on the heuristic algorithm. The heuristic intelligent algorithm is an older method that seeks optimal solutions. It was introduced into task scheduling in grid computing and improved the scheduling performance. Currently, it is widely used in task scheduling based on the Genetic Algorithm, ants algorithms and their improved algorithms. The merit of heuristic intelligence scheduling is that it can auto-adapt and optimal performance, but the disadvantage is the process of optimal very complex.

3. Agent based task scheduling

Agent technology developed originally from research into artificial intelligence. The concept was proposed in the 60s, but development laboured until the 90s. Agent technology is an effective way to describe and analyse both complex and grid systems. Characterized by intelligence and an ability to learn, it is suitable for distributed computation task scheduling. In Agent based task scheduling, the system encapsulates every grid node as an agent. This particular method reduces scheduling issues of to how to match and adjust constantly changing tasks with those of the agents, as well as how to continue assigning sub-tasks inside of the agent.

4. Economy model based scheduling algorithm

The relationship between supply and demand for resources in grid computing resource is similar to the commodity economy model. In this comparison, the resource provider is equal to the commodity supplier, and the resource consumer is equal to the commodity buyer. The basic idea of the economy model based scheduling algorithm is to set up a market mechanism to adjust the consumer’s requirements and also resource assignment. All the scheduling is based on the price release lever to optimize the system and maximize efficiency. Other schemes have also been based on auction, supermarket, marked price and bank models.

5. Other Improved Scheduling Algorithms

Besides the aforementioned introduction of a job-scheduling model, there are also some improvements to the job-scheduling algorithm. For example, in the original job scheduling algorithm, the priority level, QoS restraint, trust mechanism improvement algorithms and so on all left room for improvement.

2.3 Cloud Computing

2.3.1 Background Research

Through the above discussion, it shows that cloud computing has some commonalities with grid computing. Actually, cloud computing cannot necessarily be regarded as a new concept, but should rather be considered a subset technology from the view of its function and application. Cloud computation developed from distributed computing, parallel processing and grid computing. It is only one kind of emerging business-computing models. Until now, there has been no clear definition of cloud computing. In the context of this thesis, cloud computing is viewed to use distributed resources through all kinds of application systems or software in order to offer services to distributed users.

Normally, cloud computing is narrowly defined as the service provider who uses distributed computing resources and virtualization technology to build a data centre or supercomputer which can offer on-demand service or data storage for the user. Companies such as Amazon demand these services for their data warehouses.

Generalized cloud computation refers to the service provider through the establishment network server cluster who provides the online software service, the hardware and environment rent, the data storage, the computation analysis and so on. Examples include Salesforce.com HRM online service, and Google App and Doc.

The cloud is also regarded as the server cluster resources on the Internet. It includes hardware resources (servers, memory, CPU and so on), software resources (for example application software, integrated development environment and so on) and the network environment. Once the client computers on the Internet submit requirements, the cloud side will have thousands of computers automatically organized to provide services for client computers. This means that the clients just need to submit a request, and almost all the applications in the cloud will process it.

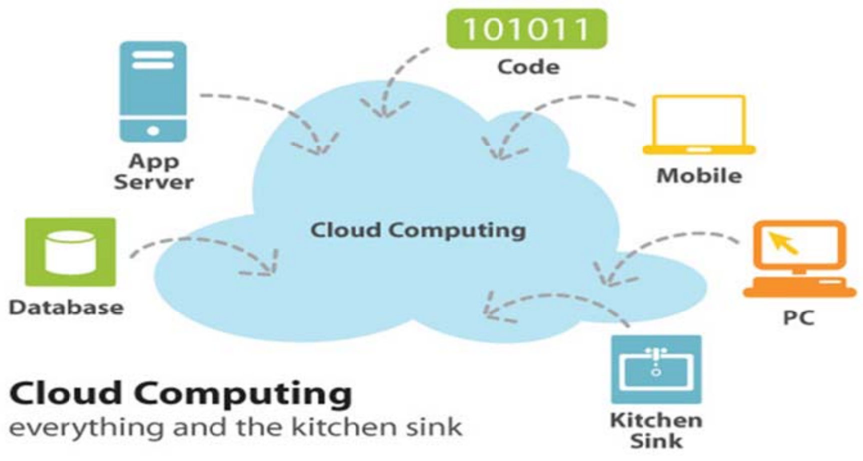


Figure 2. 3Cloud Computing Everything and the Kitchen Sink [25]

Cloud computing is projected to be one of the most important technologies in the IT sector, as illustrated by the following chart showing main IT technology ("Gartner's IT Hype Cycle"[27]).

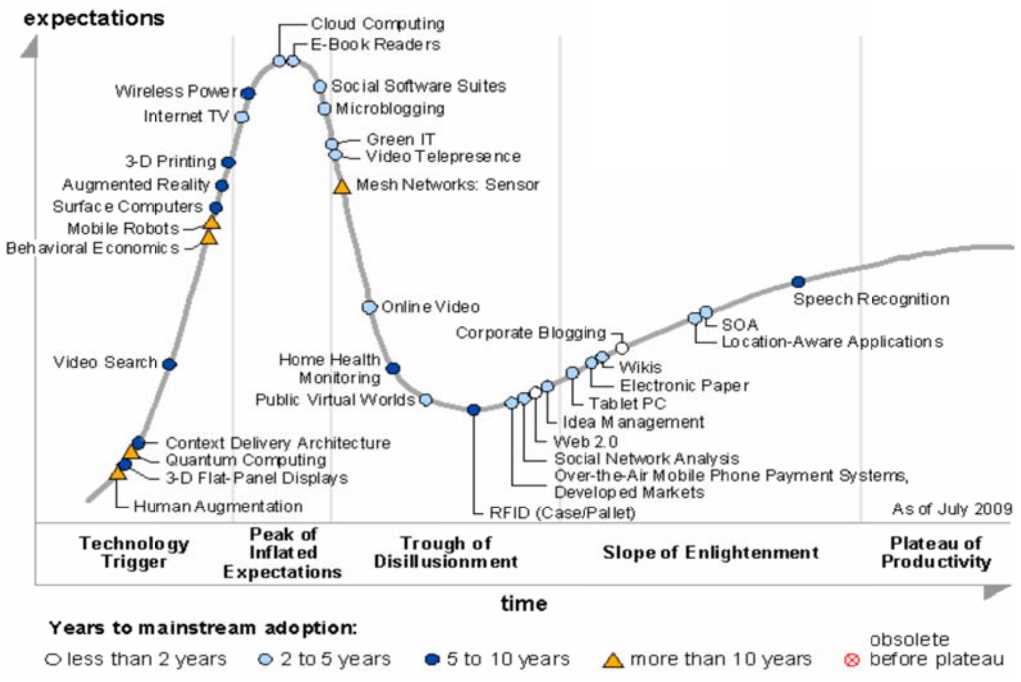


Figure 2. 4 Hype Cycle of Emerging Technologies[28]

As shown in Figure 2.4, cloud computing has been subject to increasing attention since it emerged at the end of 2007.

At the same time, the concept of cloud computing is becoming dependent on the idea of Utility Computing. The utility computing is the process of providing computing service through an on-demand, pay-per-use billing method. Utility computing is a computing business model in which the provider owns, operates and manages the computing infrastructure and resources, and the subscriber's accesses it as and when required on a rental or metered basis**[29]**. From the view of the consumer, cloud computing means the user not need to manage or pay the cost of updating the software, and also not need to purchase the hardware and other infrastructure. Actually, cloud computing advanced the Utility Model to the new level. Many different types of resources (hardware, software, memory, correspondence and so on) may immediately merge to have a specific ability in order to service customer requests. In many situations, various organizations and individuals purchase "computing ability" as a service. The following table compares the cost of traditional software and the SaaS based application[30].

Table2. 1 The General Cost of Traditional Software and SaaS Software for 200 Users[31]

The General Cost of Traditional Software and SaaS Software for 200 Users		
Scope of Comparison	Traditional Software200 Users	SaaS Software200 Users
Development Costs	The cost of about 200 individual requirements	1 software development cost
Cost of Physical Servers	200 Servers	10 Servers
Network Equipment Cost	200 Systems	5-10 Systems
IT Service Person Costs	200 Persons	2-3 Persons

The table above shows that the cloud computing based applications are more cost effective means of information system than traditional software.

2.3.2 The Definition of Cloud Computing

Cloud computing is becoming one of the most popular research areas in the computer industry. It is a user-centric service. Users store their data and applications into the remote "Cloud", which allows the user to access all the data from all their devices. Meanwhile, users can access them in a simple and pervasive way. However, the system is limited to the user and their data. If the “Cloud computing” model can be archived, then all the stored resources in the cloud could be shared easily with others.

At this point in the discussion, the central processing model appears again by pure coincidence. About 60 years ago, time sharing computers offered services for many clients. At that time, all the data and application were stored into the central server, and personal computers had to be used locally. However, the rapid development of the personal computer shifted power from the central server to the client side. Personal computers can store data and run applications by themselves.

Cloud computing is quite different from the central processing of 60 years ago. The computing structure has gone through a lot of changes. In “cloud computing”, computing resources are shared with little concern for user location or device. It is a concept in which all the resources offer services for the user centrally. The users do not need to know what kind of resources offer the service to them (such as hardware, software, and network environment) and so cloud computing is a logical central processing model for users.

Professor Mariana[32]offers the most commonly accepted definition of cloud computing: Internet-based computing, whereby shared servers provide resources, software, and data to computers and other devices on demand, as with the electricity grid. Cloud computing is a natural evolution of the widespread adoption of virtualization, service-oriented architecture and utility computing. Details are abstracted from consumers, who no longer have need for expertise in, or control over, the technology infrastructure "in the cloud" that supports them[33]. Cloud computing describes a new supplement, consumption, and delivery model for IT services based on the Internet, and it typically involves over-the-Internet provision of dynamically scalable and often virtualized resources[34]. The National Institute of Standards and Technology (NIST) provides a somewhat more objective and specific definition[35]: Typical cloud computing providers deliver common business applications online that are accessed from another Web service or software like a Web browser, while the software and data are stored on servers[35].With the rapidly development of the cloud computing, almost all the major IT companies join this area recently. Google developed IaaS and PaaS platform, Salesforce developed SaaS based application, Amazon also developed IaaS and DaaS platform. It shows as figure 2.5 as below.

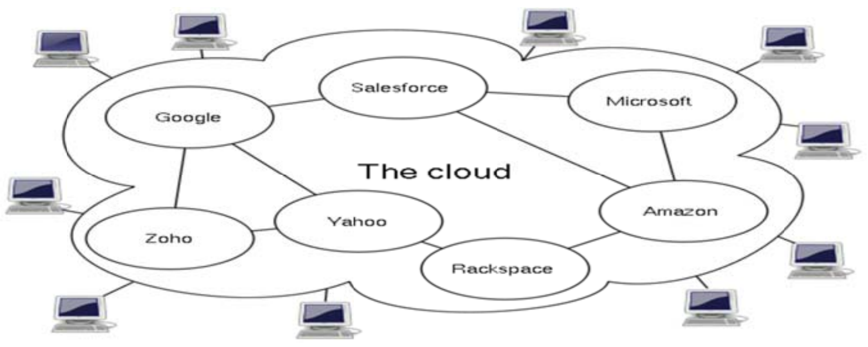


Figure 2. 5 The Main Cloud Computing Service Vendors

2.3.3 The Taxonomy of Cloud Computing

There is no clear taxonomy for cloud computing, and different views lead to different results. Most people believe that the cloud computing is a business service application based on distributed computing. Usually, it has four kinds of service models according to the different kinds of service and consumption, which are described below:

1. Software as a Service (SaaS)

The SaaS service providers deploy application software on their own server. All the application software runs on the cloud server side, and the customer can access all the applications according to their needs and requirements. The SaaS service provider charges customers according to quantity of software used and the time of usage. Essentially, SaaS supplies a software pattern through the browser to the customer.

The advantage of this kind service is that the SaaS service provider assumes all responsibility for maintenance and management of the application software, while at the same time offering all necessary hardware facilities. The costumers only need to access the Internet at their leisure. Under this kind of service provision, the customer no longer need to spent a lot of money on the hardware, software and IT service. The customer only needs to rent the service to use the corresponding hardware, software and maintenance service through the Internet. SaaS is the most effective business model for IT applications. For smaller enterprise, SaaS is the best way to access advanced technology. For example, Salesforce.com applies the HRM system to shared advanced HRM technology and management [36].Another good example of SaaS is Google Docs[37], which is the first cloud computing service vendor. It is similar to Microsoft's Office online office software. Google Docs can process and search documents, forms, and the slide, and can share documents with others through Microsoft's Office Live[38]network, and Open Social mash-ups such as Auciti's Hangout[39], MySpace and so on.

2. Platform as a Service (PaaS)

PaaS is one kind of distributed platform service, in which the development environment is regarded as a service. The service provider offers the development environment, server platform and hardware resources to the developer (customers). The developer can customize application software based on this platform, and then offer services to the end user through their own server and Internet. PaaS can offer a research platform, middleware platform, application programming development environment, database server, and experiment and trusteeship server to enterprises and individual developers. There are several good examples of PaaS, such as the Google App Engine[40] and Microsoft's Azure[41]. The Google App Engine is composed of the Python application cluster server, BigTable database and Google File System. It offers an online application service which the integrated host server and update automatically. The developer can offer service through the Internet when the developer composed and run the application software on the Google App Engine. Google provide all the necessary running time resources and platform. Microsoft's Azure offers a wide range of alternative programming tools under the Azure runtime umbrella and Joyent's Reasonably Smart[42].

3. Data as a Service (DaaS)[43]

The Journal of Map & Geography Libraries[44] offers the following definition of Data storage as a Service: DaaS is a cousin of Software as a Service[45]. Like all members of the "as a Service" (aaS) family, DaaS is based on the concept that the product, data in this case, can be provided on demand[46] to the user regardless of geographic or organizational separation of provider and consumer. Additionally, the emergence of service-oriented architecture (SOA) has rendered the actual platform on which the data resides also irrelevant[47]. This development has enabled the recent emergence of the relatively new concept of DaaS. At first DaaS was primarily used in web mash-ups, but now is being increasingly employed commercially and, less commonly, within organizations such as the UN[48]. Traditionally, most enterprises have used data stored in a self-contained repository, for which software was specifically developed to access and present the data in a human-readable form. One result of this paradigm is the bundling of both data and software needed to interpret it into a single package, sold as a consumer product. As the number of bundled software/data packages proliferated and required interaction among one another, another layer of interface was required. These interfaces, collectively known as Enterprise Application Integration (EAI), often tended to encourage vendor lock-in, as it is generally easy to integrate applications that are built upon the same foundation technology[49].The result of the combined software/data consumer package and required EAI middleware has been an increased amount of software for organizations to manage and maintain, simply for the use of particular data. In addition to routine maintenance costs, a cascading amount of software updates are required as the format of the data changes. The existence of this situation contributes to the attractiveness of DaaS to data consumers because it allows for the separation of data cost and usage from that of a specific software or platform[50].

4. Infrastructure as a Service (IaaS)

Infrastructure as a Service is a provision model in which an organization outsources the equipment used to support operations, such as storage, hardware, servers and networking components. The service provider owns the equipment and is responsible for housing, running and maintaining it. The client typically pays on a per-use basis[51]. Rather than purchasing servers, software, data centre space or network equipment, clients instead buy those resources as a fully outsourced service. The service is typically billed on a utility computing basis and amount of resources consumed (and therefore the cost) will typically reflect the level of activity[52]. Sometimes, hardware as a service (HaaS) is considered a part of IaaS.

The typical IaaS vendors are: Amazon Web Service (AWS)[53], IBM Blue Cloud[54]and Rackspace[55]. One benefit of IaaS is that it significantly reduces the cost of hardware for the user, who only needs low-cost hardware, and can rent, on demand, the necessary computing power and memory capacity.

5. The Structure of Pure Cloud Computing

The following picture shows the structure of the cloud computing. Although the relationship between the different levels of cloud computing is contentious, this thesis has rendered it as follow. From the bottom up, the lowest level is an IaaS level which offers all necessary hardware, network environment and the basic virtual OS. Based on the IaaS is a PaaS, which offers a platform for the developer and the enterprise to develop their own applications. It is also a running time environment. The upper level is SaaS and DaaS level. Both of them directly offer services to the end user. Mulit-tenant business models vitalize the new Service-oriented architecture (SOA). The DaaS offers a data warehouse to customers, which works together with SaaS to archive the final using for the end user. All mentioned pure cloud computing structure shows as figure 2.6.

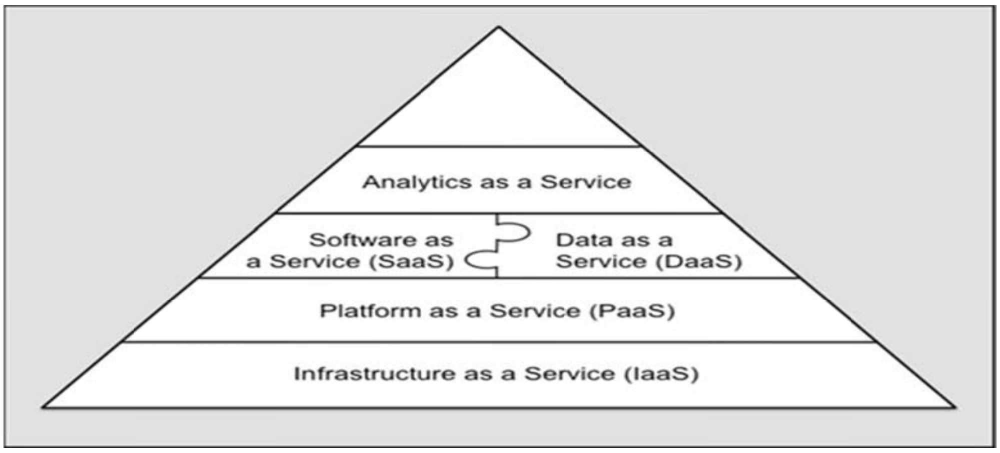

Analytics as a Service
Software as a Service (SaaS)
Data as a Service (DaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

Figure 2. 6 The Structure of Cloud Computing[56]

2.3.4 Internal Components of Cloud Computing

Based on the above discussion, this thesis will further delve into the inner working of cloud computing. Even though there only four kinds of services, the real cloud computing environment is very complex. The following chart shows in detail how to build cloud computing component by component.

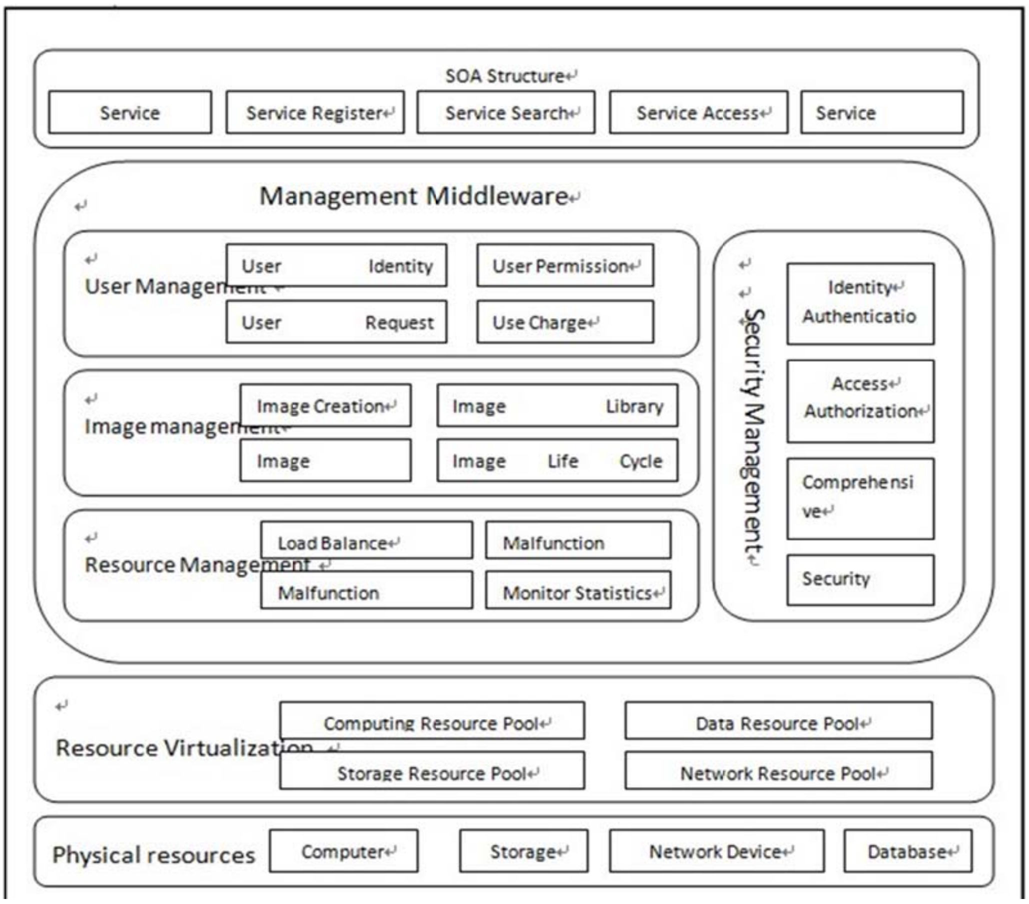


Figure 2. 7 SOA Structure[58]

Above figure shows the idealized cloud computing structure, which is composed of four layers. The bottom layer is physical machine that consisted of normal computer, storage, network device and database. In that layer, all the physical devices will reorganize and reallocated as individual resource to the upper layer. The upper layer is resource virtualization layer that tries to reassign all the resource as computing, storage, data and network resources. The resource virtualization layer offers easy interface to the upper layer within hiding details. The management middleware is belonging into the PaaS platform, which offers the API and any other necessary toolkit for developing the application in the future use. It offers all the foundational component to support the final user. The top level is face to the service management which reunion and assembles all the service based on the service flow.

2.3.5 Main Differences Between Cloud Computing and Grid Computing

Cloud Computing is the development of the Distributed Computing, Parallel Computing and Grid Computing, and should also be considered a commercial realization.

Ian Forster[60] gives a clear definition of grid computing. “Grid computing is a term referring to the combination of computer resources from multiple administrative domains to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What distinguishes grid computing from conventional high performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous, and geographically dispersed. Although a grid can be dedicated to a specialized application, it is more common that a single grid will be used for a variety of different purposes. Grids are often constructed with the aid of general-purpose grid software libraries known as middleware[61]”.

2.3.5.1 In-depth Comparison and Contrast Between Cloud Computing and Grid Computing

From the view of the definition, both of them are trying to organize all kinds of the heterogeneous IT resource as a service into the resource pool, which can in turn offer the service for outside consumption.

1. Different motivations

Cloud computing allows the user use resources transparently. Cloud computing is a broad concept; it allows the user to access all kinds of services through the Internet, based on IT resources services. This kind of service permits the user to enjoy the IT service without fully comprehending the lower level infrastructure. Grid computing emphasizes that using the IT resource is equally as simple as using water and electricity. Grid computing includes two aspects: utility computing and “the virtual supercomputer”. Utility computing, or on demand computing, is similar between grid computing and cloud computing. Both of them are provide online services based on distributed computing resources. “The virtual super computer” refers to the fact that grid computing connects individual computers into a massive computing resource to provide supercomputing abilities beyond the scope of individual computing resources.

2. Different purposes

The purpose of grid computing is to try to use any resources regardless of how far away they are. Specific grid computing middleware decomposes a huge project into innumerable mutually independent, not too related sub-duties, which are then processed by computing nodes. Even if one of the nodes has a problem and cannot return results on time, it does not affect the result of whole project. For example, if a computing node crashes suddenly, its uncompleted tasks can be assigned by the task scheduling system to other nodes for completion. From the view of the scalability, the core value of grid computing is job scheduling. Generally speaking, grid computing is designed to complete some specific task. In contrast, cloud computing is designed for the general applications. Cloud computing provides one kind of service or resource for outside consumption in order to finish one special task. For example, some users may need to use an application to deploy their own application into the resource pool, but they will not submit the whole task to the cloud for computation. The main difference between the purposes of grid computing and cloud computing is that grid computing tries to use all kinds of resources to finish one application, and the cloud computing tries to use all kinds of resources to finish all kinds of applications.

3. Different resource method of assign

Grid computing is a resource pool. For the resource user, they only need to submit the job to the grid, and they do not need to know how it works and which physical node will be used. They only need to follow the specific form and submit the job to the system, then wait for the grid system to return the results. The grid job scheduling system searches the resources and automatically matches them to tasks before forwarding the component job to an idle physical node. This cycle repeats until the task is completed. Although the grid can process the job using different physical machines, it needs the user to write the parallel algorithm and divide the whole job into sub-jobs to run under different physical nodes. This process is quite complex, which is one reason why grid computing systems are designed for very specific purposes. Cloud computing integrates all the physics machine resources through virtualization in order to achieve automatic flexibility and fault tolerance. All the applications on the cloud cannot go beyond the physical node's upper limit. In terms of control, cloud computing regards all IT resources as one resource pool and the different chip machines are classified into a different resource pool.

2.4 Cloud Resource Management

Scheduling onto the cloud is NP-complete, so there is no ideal scheduling algorithm for all cloud computing systems. An alternative is to select an appropriate scheduling algorithm to use in a given cloud environment because of the characteristics of the tasks, machines and network connectivity. Job and resource scheduling is one of the key research areas in cloud computing. The goal of scheduling is to achieve the highest possible system throughput and to match the application needed with the available computing resources.

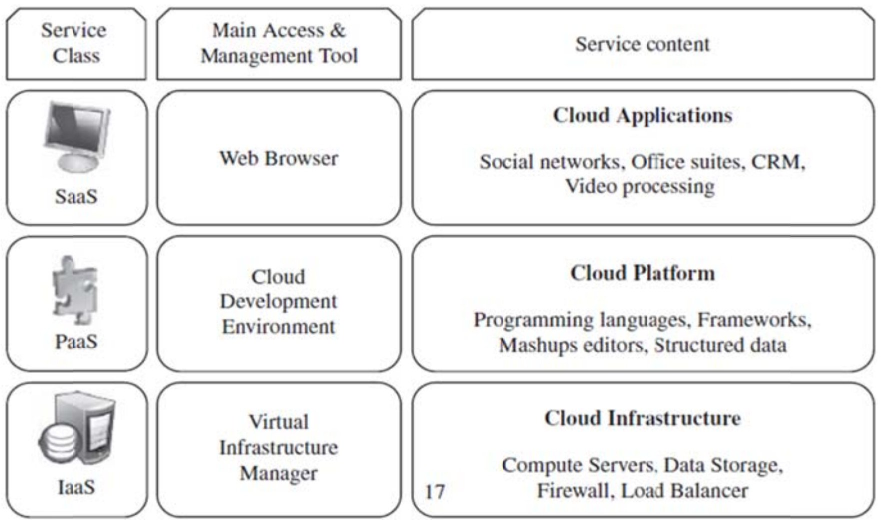
2.4.1 Main Strategies of Cloud Resource Management

Technologies such as cluster, grid, and now, cloud computing, have all aimed at allowing access to large amounts of computing power in a fully virtualized manner by aggregating resources and offering a single system view. In addition, an important aim of these technologies has been delivering computing as a utility. Utility computing describes a business model for on-demand delivery of computing power. Consumers pay providers based on usage ("pay-as-you-go"), similar to the way in which we currently obtain services from traditional public utility services such as water, electricity, gas, and telephony[62].

Virtualization technology is the key of IaaS clouds because it gives providers a more flexible and generic way of managing their resources. Thus, virtual infrastructure (VI) management—the management of virtual machines distributed across a pool of physical resources—becomes a key concern when building an IaaS cloud and it poses a number of challenges. Like traditional physical resources, virtual machines require a fair amount of configuration, including preparation of the machine's software environment and network configuration[63].

However, in a virtual infrastructure, this configuration must be done on-the-fly, with as little time between the time the VMs are requested and the time they are available to the user. This is further complicated by the need to configure groups of VMs that will provide a specific service (e.g., an application requiring a Web server and a database server). Additionally, a virtual infrastructure manager must be capable of allocating resources efficiently while considering an organization's goals (such as minimizing power consumption and other operational costs) and reacting to changes in the physical infrastructure.

2.4.2 The Taxonomy of Cloud Resource Management



Service Class	Main Access & Management Tool	Service content
SaaS	Web Browser	**Cloud Applications** Social networks, Office suites, CRM, Video processing
PaaS	Cloud Development Environment	**Cloud Platform** Programming languages, Frameworks, Mashups editors, Structured data
IaaS	Virtual Infrastructure Manager	**Cloud Infrastructure** 17 Compute Servers, Data Storage, Firewall, Load Balancer

Figure 2. 8 The Taxonomy of Cloud Resource Management[64]

The above figure demonstrates that different classes of cloud computing focus on managing different resources. At present, cloud computing resource management is a key problem facing every large IT company. To date, there are no uniform standards. Resources schedules and assigned in the IaaS level include all the abstract hardware resources, such as CPUs, memory capacity, hard drive capacity, broadband and so on.

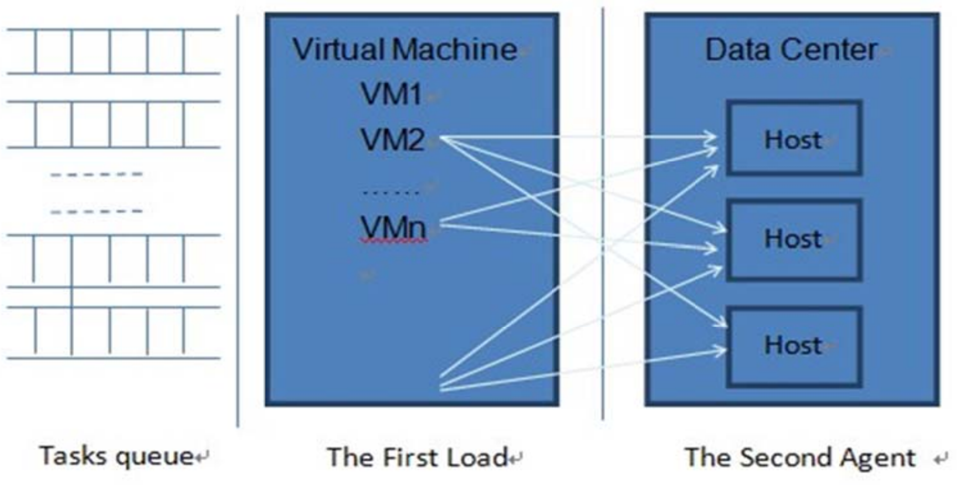


Figure 2. 9 Load Balance of IaaS Cloud Computing

From the above figure, it is clear that the scheduling of IaaS is a way of balancing loads among the virtual machines and data centre. Any VM could compose by different resource. The VM can on demand divided as difference ability size according to the requirement. The resource scheduling can assign the right task to the right resource to perform it. The following cases are several solutions to scheduling IaaS level resources.

1. The IBM Tivoli

IBM's Tivoli products are one of the products in the Blue Cloud program, which uses policy-based resource scheduling, storage and systems management solutions. It provides an integrated view of the management and optimization of IT systems. This product uses Websphere Application Server to provide the availability for access data centre resource or scheduling and booking the resources. Virtualization technology is an important feature of this platform, which focuses mainly on hardware-level virtualization, and open source software is used to achieve virtualization. The hardware-level virtualization servers use IBM p series servers to access the hardware logical partition. The IBM Enterprise Workload Manager manages the logical partition of CPU resource. This method can reasonably allocate appropriate resources to each logical partition, and using the virtual machine to manage the resources can fully simulate the implementation of the hardware. At the same time, it allows users to have single computer operation experience. It can also perform tasks on multiple operating systems.

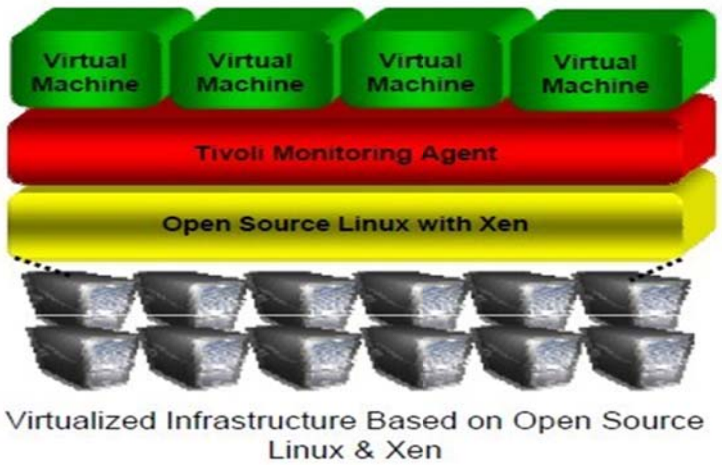

Virtual Machine
Virtual Machine
Virtual Machine
Virtual Machine
Tivoli Monitoring Agent
Open Source Linux with Xen
Virtualized Infrastructure Based on Open Source Linux & Xen

Figure 2. 10 IBM High Performance Solution on Demand (HiPODS)[66]

2. Dryad

Dryad is a research project from Microsoft Research Lab. It is mainly used to provide a distributed parallel computing platform. It is not only a programming model but also an efficient task-scheduling model. Dryad is designed to be scalable to any size of cluster computing platform, whether it is a single multi-core computer or a cluster consisting of multiple computers, even with thousands of computer data centres. Such as cloud computing, can benefit from the job queue creating a strategy for modelling, programming for distributed parallel computing.

As a task-scheduling model, each Dryad task is represented as a directed acyclic graph, and each node represents a program to be executed.

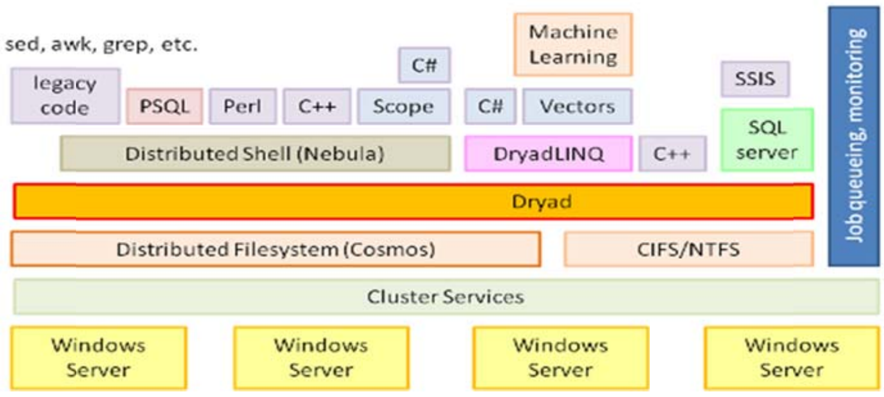


Figure 2. 11 Dryad Platform[68]

Users must build their own tasks in the task node in the Dryad platform. Each task is composited by the processing and the related data. When the Task Manager obtains the acyclic graph, it will prepare for scheduling in the input channel. Once a machine is available, the Task manager will obtain a list of available computers, and then uses a process to scheduling the program[70].

3. Google MapReduce

MapReduce[71] is a software framework introduced by Google in 2004 to support distributed computing on large data sets on clusters of computers. Parts of the framework are patented in some countries.

The framework is inspired by the map and reduces functions commonly used in functional programming, although their purpose in the MapReduce framework is not the same as their original forms.

MapReduce is a framework for processing huge datasets on certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster (if all nodes use the same hardware) or as a grid (if the nodes use different hardware).

Computational processing can occur on data stored either in a file system (unstructured) or within a database (structured)[72].

"Map" step: The master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node.

"Reduce" step: The master node then takes the answers to all the sub-problems and combines them in some way to get the output – the answer to the problem it was originally trying to solve.

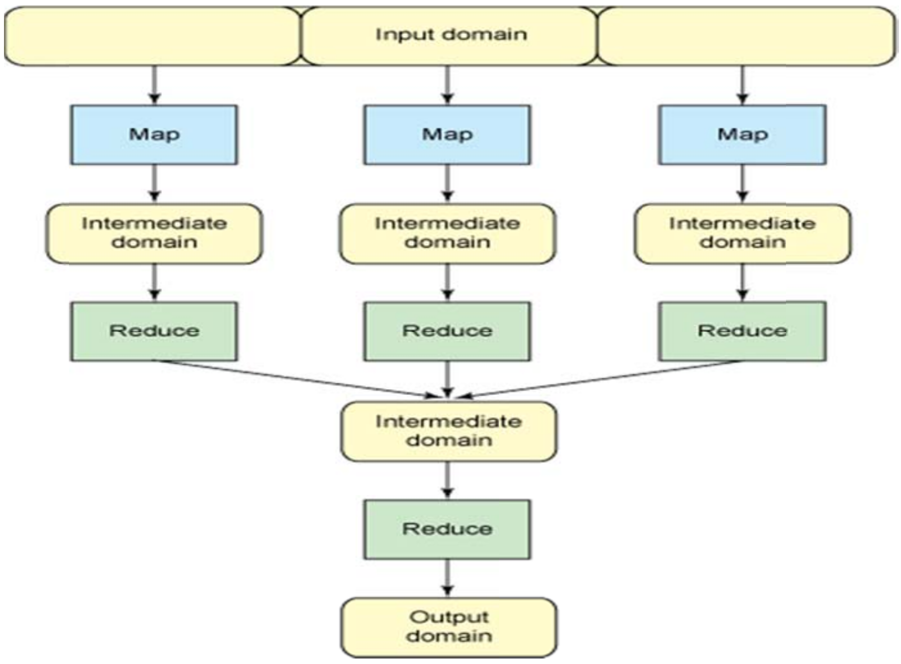


Figure 2. 12 General MapReduce Processing

In the Map Reduce programming model, all the concurrent processing details, the details of data distribution, and load balancing are abstracted into a library. Through the MapReduce interface, users can automatically execute the entire task concurrently and distributed. In the MapReduce programming model, procedure is abstracted into three roles: master, worker, and user.

The master is the central control system, and is mainly responsible for data division, task allocation, load balancing, fault tolerance, and so on. The worker is responsible for processing and computing tasks received from the master, as well as data communication. In order to do so, the worker implements the two-stage map and reduces functions, by modelling abstract large-scale data into the two stages. The map stage maps all the data as key-value pairs. The reduce stage simplifies and merges the key-value pairs. As the master program, master is responsible for selecting the tasks assigned to idle workers. The task of the worker is to parse the key-value pairs and implement map operations, and map the results of the key to the cache on the local disk and return the address to the master. The worker obtains the address of the intermediate results from the master's key, reads the data, sorts and simplifies keys, and then returns the results to the user program. Figure 2.13 shows the overall flow of a MapReduce operation in implementation.

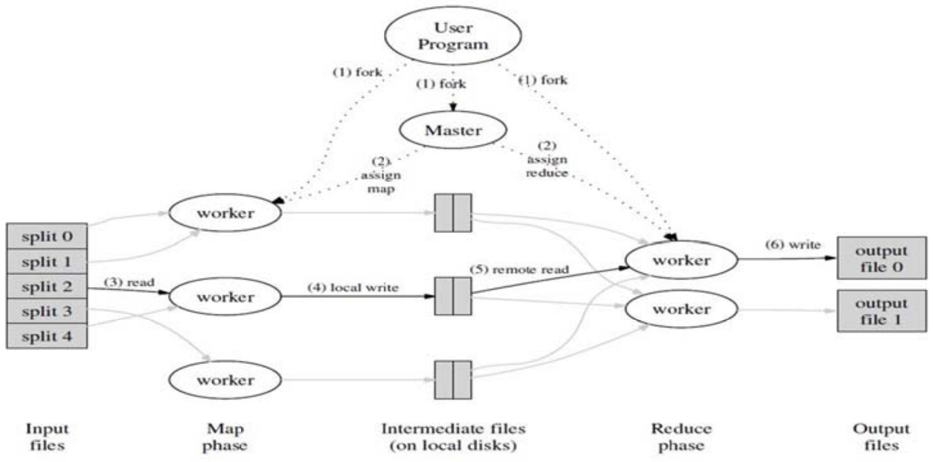


Figure 2. 13 Overview of MapReduce Process[73]

Jeffrey Dean and Sanjay Ghemawat[74] have showed the execution process from step 1 to 6 as below:

1. The MapReduce library in the user program first splits the input files into M pieces of typically 16 megabytes to 64 megabytes (MB) per piece (controllable by the user via an optional parameter). It then starts up many copies of the program on a cluster of machines.
2. One of the copies of the program is special – the master. The rest are workers that are assigned work by the master. There are M map tasks and R reduces tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.
3. A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined Map

function. The intermediate key/value pairs produced by the Map function are buffered in memory.

4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master, who is responsible for forwarding these locations to the reduce workers.
5. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same reduce task. If the amount of intermediate data is too large to fit in memory, an external sort is used.
6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key en- countered, it passes the key and the corresponding set of intermediate values to the user's Reduce function. The output of the Reduce function is appended to a final output file for this reduces partition.
7. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the MapReduce call in the user pro- gram returns back to the user code

There are two advantages of MapReduce: first, MapReduce is a distributed computing framework, and can not only be used to deal with large-scale data computing, but it also shields the user from trivial details. Second, Map/Reduce is very flexible. Whenever a new server is added, a Map/Reduce cluster can get a similar computing capacity. Other distributed computing frameworks in the past were very rigid by comparison. The biggest shortcoming of MapReduce, however, is that it cannot adapt to requests from real-time applications.

4. Windows Azure Fabric Controller

Windows Azure is a cloud computing operating system, including operating systems, application infrastructure services and components, mainly for the management of virtualized computing resources and the intelligent distribution of tasks[76].The Azure Fabric Controller (FC) is the part of the Windows Azure platform that monitors and manages servers and coordinates resources for software applications. The program functions as the kernel of the

Azure operating system. It provides, stores, delivers, monitors and commands the virtual machines (VMs) and physical servers that make up Azure[77].

All Windows Azure applications and all of the data in Windows Azure Storage live in some Microsoft data centre. Within that data centre, the set of machines dedicated to Windows Azure is organized into a fabric. Figure 2.14 shows how this looks.

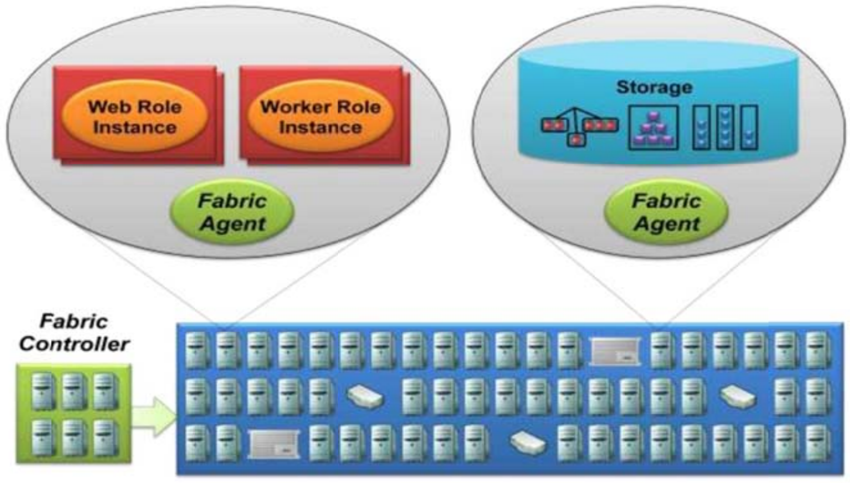


Figure 2. 14 The Controller Interacting with Windows Azure Applications via Fabric Agent[78].

As the figure shows, the Windows Azure Fabric consists of a (large) group of machines, all of which are managed by software called the fabric controller. The fabric controller is replicated across a group of five to seven machines, and it owns all of the resources in the fabric: computers, switches, load balancers, and more. Because it can communicate with a fabric agent on every computer, it is also aware of every Windows Azure application in this fabric. (Interestingly, the fabric controller sees Windows Azure Storage as just another application, and so the details of data management and replication are not visible to the controller.)[79]. The fabric controller's awareness depends on a configuration file that is uploaded with each Windows Azure application. This file provides an XML-based description of what the application needs: how many Web roles, how many worker roles, and so on, and when the fabric controller receives this new application, it uses this configuration file to determine how many role VMs to create[79].

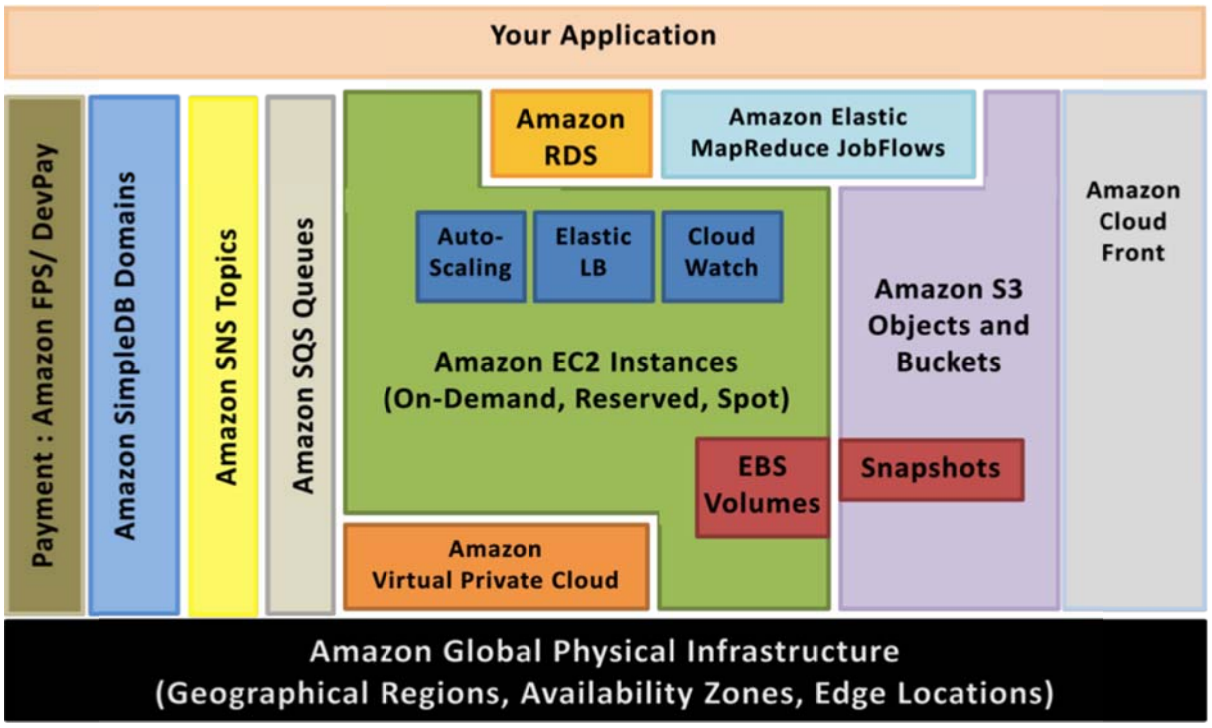
The fabric controller creates and then monitors each of these VMs. For example, if an application requires five Web role instances and one of them dies, the fabric controller will automatically restart a new one. Similarly, if the machine a VM is running on dies, the fabric controller will start a new instance of the web or worker role in a new VM on another machine, re-setting the load balancer as necessary to indicate this new machine[79].

While this might change over time, the fabric controller in the Windows Azure CTP maintains a one-to-one relationship between a VM and a physical processor core. Because of this, performance is predictable— each application instance has its own dedicated processor core. It also means that there is no arbitrary limit on how long an application instance can execute tasks. A web role instance, for example, can take as long as it needs to handle a request from a user, while a worker role instance can compute the value of pi to a million digits if necessary. Developers are free to do what they think is best[79].

5. Amazon Elastic MapReduce & Hadoop

Amazon Web Services (AWS) [80] is a collection of remote computing services (also called web services) that together make up a cloud computing platform, offered over the Internet by Amazon.com. The most central and well-known of these services are Amazon EC2 and Amazon S3. The most brilliant example of applying AWS is the New York Times using 100 Amazon EC2 instances and a Hadoop application to process 4TB of raw image TIFF data (stored in S3) into 11 million finished PDFs in the space of 24TB.

Most application needs computer, storage, messaging, payment, distribution, scale and analytics. The AWS cloud computing offer all the necessary component to address those task. The bottom lever offers physical infrastructure, especially worldwide region physic device to offer basic computing and storage. Based on the virtualization technology, the AWS off on demand EC2 instance, functions as a virtual private server. At the same time, it also provides audit, payment, security and other functionality components. The logical model shows as figure 2.15



Your Application
Payment : Amazon FPS/ DevPay
Amazon SimpleDB Domains
Amazon SNS Topics
Amazon SQS Queues
Amazon RDS
Amazon Elastic MapReduce JobFlows
Auto-Scaling
Elastic LB
Cloud Watch
Amazon EC2 Instances (On-Demand, Reserved, Spot)
EBS Volumes
Amazon Virtual Private Cloud
Amazon S3 Objects and Buckets
Snapshots
Amazon Cloud Front
Amazon Global Physical Infrastructure (Geographical Regions, Availability Zones, Edge Locations)

Figure 2. 15The AWS Infrastructure Platform[81]

Since early 2006, Amazon Web Services has provided online services for other web sites or client-side applications. Most of these services are not directly exposed to end users, but instead offer functionality that other developers can use.

Amazon Elastic MapReduce is an important part of AWS. It is a web service that makes it easy to process large amounts of data efficiently. Elastic MapReduce uses Hadoop processing combined with several other AWS products to do such tasks as web indexing, data mining, log file analysis, machine learning, scientific simulation, and data warehousing.

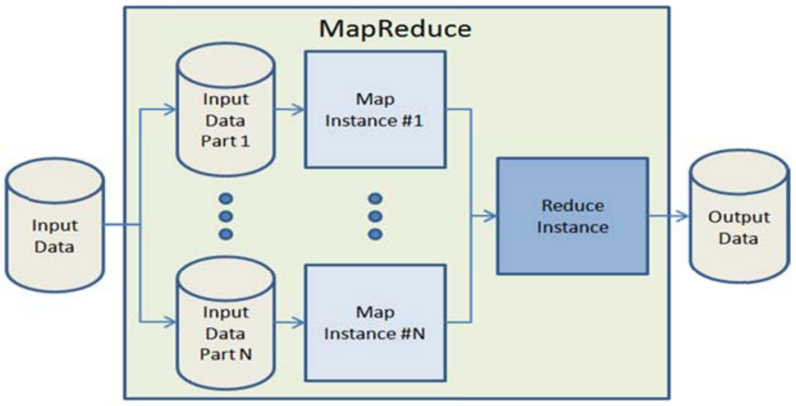


Figure 2. 16High Level Architecture of Hadoop MapReduce (A)[82]

Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license[84]. It enables applications to work with thousands of nodes and petabytes of data. Google's MapReduce and Google File System (GFS) papers inspired Hadoop.

Hadoop Common forms the Hadoop, and provides access to the file-systems supported by Hadoop. The Hadoop Common package contains the necessary JAR files and scripts needed to start Hadoop. The package also provides source code, documentation, and a contribution section that includes projects from the Hadoop community.

For effective scheduling of work, every Hadoop compatible file-system provides location awareness: the name of the rack (more precisely, of the network switch) where a worker node is located. Hadoop applications can use this information to run work on the node where the data is located, and, failing that, on the same rack/switch, thus reducing backbone traffic. The Hadoop Distributed File System (HDFS) uses this process when replicating data to try to keep different copies of the data on different racks. The goal is to reduce the impact of a rack power outage or switch failure so that even if these events occur, the data may still be readable[85].

A small Hadoop cluster will include a single master and multiple worker nodes. The master node consists of a jobtracker, tasktracker, namenode, and datanode. A slave or worker node consists of a datanode and tasktracker, though it is possible to have data-only worker nodes, and compute-only worker nodes; these are normally only used in non-standard applications. Hadoop requires JRE 1.6 or higher. The standard startup and shutdown scripts require SSH to be set up between nodes in the cluster.

In a larger cluster, the HDFS file-system is managed through a dedicated namenode server to host the file system index, and a secondary namenode that can generate snapshots of the namenode's memory structures, thus preventing file system corruption and reducing loss of data. Similarly, a standalone jobtracker server can manage job scheduling. In clusters where the Hadoop MapReduce engine is deployed mitigation an alternate file system, the namenode, secondary namenode and datanode architecture of HDFS is replaced by the file system-specific equivalent.

The MapReduce framework is composed by two kinds of scheduling services, which are JobTracker and TaskTracker. JobTracker is the only master controller to schedule and manage TaskTracks. JobTracker allocate the task of Map and Reduce to the idle TaskTrackers. Meanwhile, the JobTracker will watch performance of all the TaskTrackers. The TaskTracker is a slave node to perform all the tasks. Once some of the TaskTracker failed, the JobTracker will reallocate others to take their place. The HDFS is the master and slave architecture. It has one NameNode and many other DataNode. The NameNode is in charge of meta data and DataNode store the pure data. As the same mechanism as MapReduce, the DataNode keep mulitcopies in different data blocks. It shows as figure 2.18.

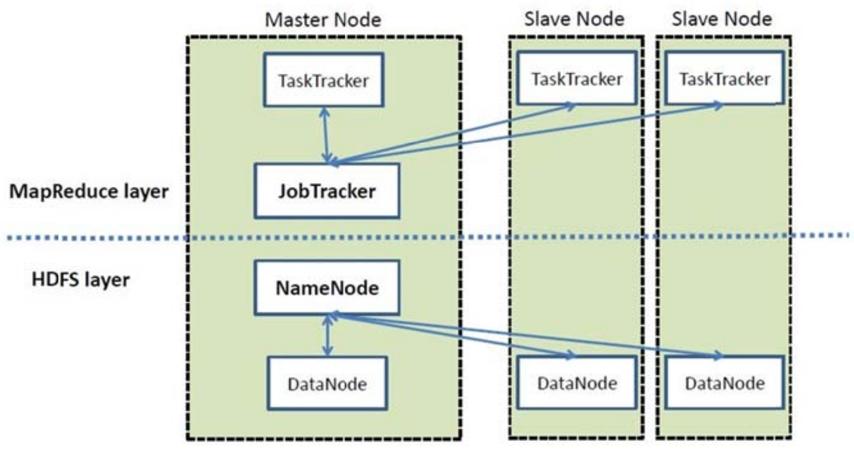


Figure 2. 17High Level Architecture of Hadoop MapReduce (B)[86]

HDFS is the file system component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, faithfulness to standards was sacrificed in favour of improved performance for the applications at hand[88]. HDFS is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single datanode. A cluster of datanodes form the HDFS cluster. The situation is typical because each node does not require a datanode to be present. Each datanode serves up blocks of data over the network using a block protocol specific to HDFS.

Because Hadoop can work directly with any distributed file system, Amazon has developed its own file-system called Amazon Simple Storage Service (S3) to target the clusters hosted in the Amazon Elastic compute cloud sever-on-demand infrastructure. Many other companies and organizations are using Hadoop to run their large distributed computations, such as Facebook, AOL, Apple, Yahoo! and its Yahoo! Search Map, and so on.

There is a limitation in AWS Elastic MapReduce. The number of machine instances that clients can run at the same time on AWS infrastructure, and therefore the number or instances in the AWS map-reduce cluster, is limited to 20. This is a problem still waiting to be addressed by Amazon.

6. Eucalyptus[89]

Eucalyptus stands for "Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems", and is a software platform for implementing private cloud computing on computer clusters. The development of Eucalyptus software is sponsored by Eucalyptus Systems, a venture-backed start-up[90]. There is currently an open-core enterprise edition and an open-source edition, and it exports a user-face interface compatible with the Amazon EC2 and S3 services, but the platform is modularized so that it can support a set of different interfaces simultaneously. Eucalyptus also works with most currently available Linux distributions including Ubuntu, Red Hat Enterprise Linux (RHEL), CentOS, SUSE Linux Enterprise Server (SLES), openSUSE, Debian and Fedora. Moreover, it can also host Microsoft Windows images, and use a variety of virtualization technology including VMware, Xen and KVM hypervisors to implement the cloud abstractions that it supports.

The Eucalyptus cloud computing platform has five high-level components: cloud controller (CLC), cluster controller (CC), walrus, storage controller (SC) and node controller (NC)[91].

The cloud controller (CLC) is responsible for exposing and managing the underlying virtualized resources (machines (servers), network, and storage) via user-facing APIs. Currently, the CLC exports a well-defined industry standard API (Amazon EC2) and via a Web-based user interface[92] .

Walrus implements scalable "put-get bucket storage." The current implementation of Walrus is interface compatible with Amazon's S3 (a get/put interface for buckets and objects), providing a mechanism for persistent storage and access control of virtual machine images and user data[92].

The cluster controller (CC) controls the execution of virtual machines (VMs) running on the nodes and manages virtual networking between VMs, and between VMs and external users[92].

The storage controller (SC) provides block-level network storage that can be attached dynamically by VMs. The current implementation of the SC supports semantics of Amazon Elastic Block Storage (EBS).

The node controller (NC) controls VM activities through the functionality of a hypervisor, including the execution, inspection, and termination of VM instances[92].

The eucalyptus has six components inside:

1. The Cloud Controller (CLC) is a Java program that offers EC2-compatible interfaces, as well as a web interface to the outside world. In addition to handling incoming requests, the CLC acts as the administrative interface for cloud management and performs high-level resource scheduling and system accounting. The CLC accepts user API requests from command-line interfaces like euca2ools or GUI-based tools like the Eucalyptus User Console and manages the underlying compute, storage, and network resources. Only one CLC can exist per cloud and it handles authentication, accounting, reporting, and quote management.

2. Walrus, also written in Java, is the Eucalyptus equivalent to AWS Simple Storage Service (S3). Walrus offers persistent storage to all of the virtual machines in the Eucalyptus cloud and can be used as a simple HTTP put/get storage as a service solution. There are no data type restrictions for Walrus, and it can contain images (i.e., the building blocks used to launch virtual machines), volume snapshots (i.e., point-in-time copies), and application data. Only one Walrus can exist per cloud.

3. The Cluster Controller (CC) is written in C and acts as the front end for a cluster within a Eucalyptus cloud and communicates with the Storage Controller and Node Controller. It manages instance (i.e., virtual machines) execution and Service Level Agreements (SLAs) per cluster.

4. The Storage Controller (SC) is written in Java and is the Eucalyptus equivalent to AWS EBS. It communicates with the Cluster Controller and Node Controller and manages Eucalyptus block volumes and snapshots to the instances within its specific cluster. If an instance requires writing persistent data to memory outside of the cluster, it would need to write to Walrus, which is available to any instance in any cluster.

5. The VMware Broker is an optional component that provides an AWS-compatible interface for VMware environments and physically runs on the Cluster Controller. The VMware

Broker overlays existing ESX/ESXi hosts and transforms Eucalyptus Machine Images (EMIs) to VMware virtual disks. The VMware Broker mediates interactions between the Cluster Controller and VMware and can connect directly to either ESX/ESXi hosts or to vCenter Server.

6. The Node Controller (NC) is written in C and hosts the virtual machine instances and manages the virtual network endpoints. It downloads and caches images from Walrus as well as creates and caches instances. While there is no theoretical limit to the number of Node Controllers per cluster, performance limits do exist[93].

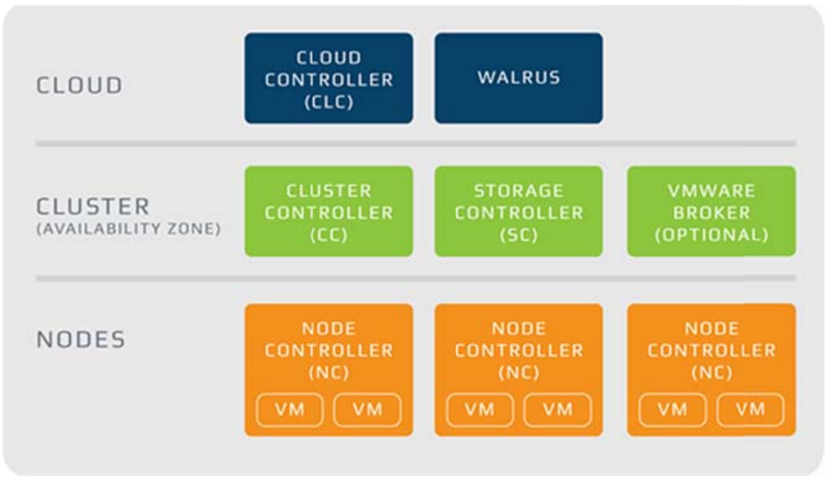


Figure 2. 18 Eucalyptus components

Each high-level system component has its own Web interface and is implemented as a stand-alone Web service. This has two major advantages. First, each Web service exposes a well-defined language-agnostic API in the form of a WSDL document containing both the operations that the service can perform and the input/output data structures. Second, Eucalyptus leverages existing Web-service features such as security policies (WSS) for secure communication between components and relies on industry-standard web-services software packages.

The famous Ubuntu Enterprise Cloud (UEC) is based on both on Eucalyptus and Intel® Cloud Test bed referenced UEC' architecture[94].

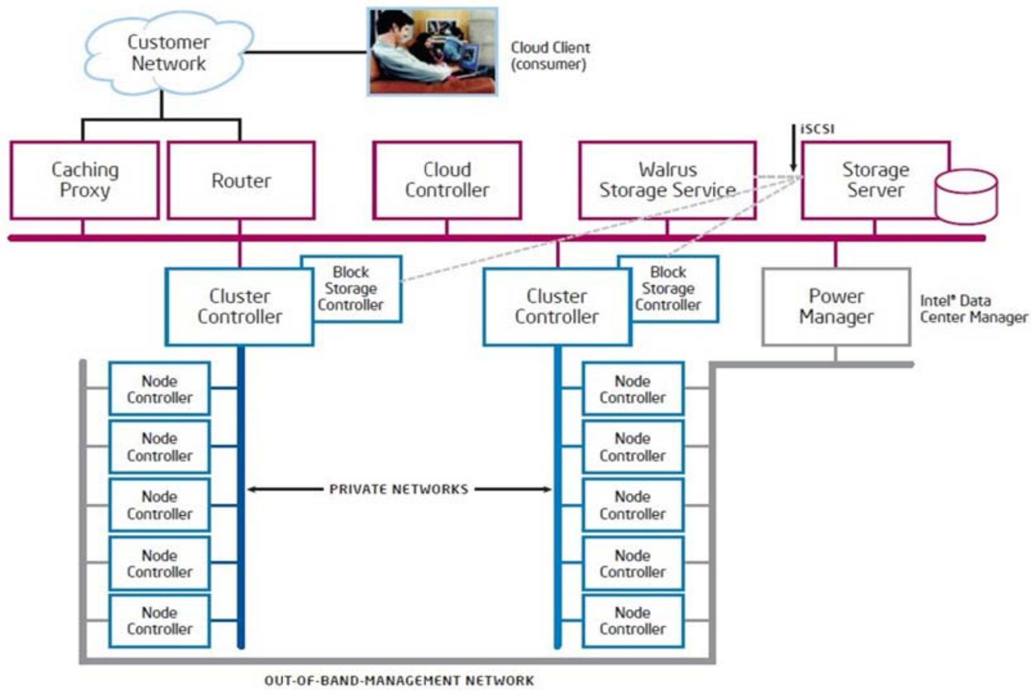


Figure 2. 19Reference Cloud Design Using Eucalyptus in Intel® Cloud Test Bed

2.4.3 Key Cloud Resource Management Technology

Cloud computing systems are dependent on a wealth of technology, key of which are the programming model, data management, data storage technologies, virtualization, cloud computing platform management technology. The following section will detail these technologies, and the ensuing discussion will use Google as an example.

2.4.3.1 Virtualization Technology

Virtualization is a software technology that divides a physical resource (such as a server) into virtual resources called virtual machines (VMs).Virtualization allows users to consolidate physical resources, simplify deployment and administration, and reduce power and cooling requirements. While virtualization technology is most popular in the server world, virtualization technology is also being used in data storage such as Storage Area Networks, and inside of operating systems such as Windows Server 2008 with Hyper-V[95].

2.4.3.2 Programming Model

MapReduce is a Google developed java, Python, C + + programming model, which is a simplified programming model and an efficient distributed task scheduling model for large-scale data sets (greater than 1TB) for parallel computing. A strict programming model simplifies programming in cloud computing environments. The idea behind the MapReduce model is to decompose problems into a Map (mapping) and Reduce (simplification) approach. First, through the Map program, it cuts the program into non-related data blocks and then allocates (schedules) them to a large number of computers for processing. This step is distributed computing. In the second step, the reduce program outputs the results of the process.

2.4.3.3 The Distribution of Mass Data Storage Technology

Cloud computing systems consist of a large number of servers with a large number of users, so the cloud computing systems require distributed storage, as a way to ensure the reliability of data and avoid localized geographic problems from jeopardizing the safety and security of their data. The data storage system in cloud computing systems has been widely used in Google's GFS and HDFS, which were developed by the Hadoop team, based on the open source implementation of GFS.

GFS (Google File System) is a scalable distributed file system for applications that access vast amounts of large-scale, distributed data. The design of GFS is necessarily different from a traditional file system for it features large-scale data processing designed for Google Apps. It runs on inexpensive commodity hardware, but can provide fault tolerance. A GFS cluster consists of a master server and a large number of chunk servers, constituted by the many clients who access the system. A master server stores all the file system’s meta-data, including namespace, access control information, the mapping from files to the block and the current position. It also controls system-wide activities such as managing block leases, garbage collection of orphan blocks, and block server migration between the servers. The main server communicates with chunk servers through the Heart Beat. The master server also passes command and collects its status. The GFS in the 64MB file is cut into blocks and placed into redundant storage. Each data owner then has more than 3 backups in the system at any given time. The client and main server for exchange only metadata operations, all data blocks of communication and contact the server directly, which greatly improves the efficiency of the system, thereby preventing an overload the main server.

2.4.3.4 Massive Data Management Technology

As cloud computing must process and analyze a massive amount of distributed data, the data management techniques must be able to efficiently manage large amounts of data. To that end, the technology underlying the cloud computing data management system is mainly Google BigTable (BT) and HBase. BT is a large distributed database, which is based on the GFS, Scheduler, Lock Service, and MapReduce. Unlike traditional relational databases, BT is able to handle all data as an object, forming a huge table for the distribution of large-scale structured data storage.

2.5 Summary

In summary, in a cloud computing environment, there are many issues that must to be resolved in order to meet user needs and improve resource allocation as well as other aspects of performance. Since the ultimate goal of cloud computing is applicability to enterprise business, creating a healthy and effective economic system is critical. In-depth study and analysis of commercial needs boils down to one key point—that an effective mechanism for resource scheduling is the key to attracting users.

Currently, most IaaS projects are built to create a private cloud, and subsequently there is no uniform standard for resource cost and payment. This is made more difficult by the reality that there is no single IaaS project to consider in a public cloud IaaS environment. Multiple resource providers integrate cloud computing services in order provide a variety of resources through a unified front. Each resource provider seeks to maximize profit, and the competition among them directly affects resource prices. Therefore, all actors in the cloud face the ongoing problem of a lack of reasonable and effective price and trading strategies. Certain business models may hold an answer, as their understanding of the economy and society, and consumers and producers translates to the cloud environment. Therefore, introducing economic theory to resolve resource pricing issues and modes of operation in the cloud is a reasonable solution.

Chapter 3 Economics and Cloud Computing Resource Management

Stable clouds rely on a well designed and built computing architecture, which can be likened to the skeleton of a biological system. Building this core is complicated by the reality of an integrated computing resource environment, which must integrate resource management, information management, data management, service quality assurance, safety and other basic functional modules. Indeed, server system architecture has evolved since early distributed computing into systems such as the five-hour glass architecture based Globus[96], the OGSA (Open Grid Services Architecture)[97], the WSRF (OASIS Web Services Resource Framework)[98] and Vega Grid architecture[99]. However, some of the existing parallel and distributed computing resource management systems cannot fully support the above-mentioned characteristics of cloud computing resources and are not well adapted to distributed resource management either. For example the Load Sharing Facility (LSF) [100]cannot support heterogeneous grid resource management, and The DQS (Distributed Queuing System)[101] is poorly adapted to dynamic and autonomic resource management. These limitations however, are now subject to market-driven innovation. That is to say, more so that with previous technology, the evolution of the cloud computing technology will respond to social needs and economic laws.

3.1 Survey of Economic Theories Based on the Grid Resource Management Project

The purpose of grid computing, and by extension cloud computing, is to optimize the use of computing resources. Grid computing technology and architecture is more effective at this compared with traditional grid computing and network computing because of an infusion of economic thought.

Market economics offers an effective way of managing grid resources, as is demonstrated by its application in several commercial successes. For example, G-Commerce[102]is a research project of University of Tennessee, and it uses market economics to allocate fluctuating resources in the grid. The project leader proposed that the relative value of resources is based on changes in supply and demand change. In the project they included a price equation for CPU resources and disk resources, and identified price adjustment as an issue. Zeta Grid [103] is a platform out of the IBM B¨oblingen laboratory in Germany, and it can create a large-scale trusted grid computing environment through the use of a commercial grid architecture[104]originally proposed by Chris Kenyon. The grid takes advantage of hierarchical layers and their properties, and it contracts price-related resources to interact with the user.

Another example, the literature offers a several models for managing grid resources, the resource binding model[105], analyses the resource binding description, pricing, and resource optimization of binding, and binding processes based on the resource requirements and QoS requirements. This model is effective for binding resources to the corresponding tasks. Yun Yang 2007[106]proposes a mechanism based on supply and demand balance for storage resources, Hao et al 2008[107]proposed a banking-based model to schedule and allocate resources, while another model,[108] proposes a market-based macroeconomic framework. The computing trust value method [109] based on the dynamic changing rule function and a constructed grid trust model based on behaviour. Because of the demands of a multi-application environment, the model improved the grid economic model Deadline and Budget Constrained (DBC) [110]Scheduling algorithm and proposed time optimization, cost optimization and time-cost trade-off optimization as his goal scheduling algorithms (Trust DBC) based on trust. A management method [111]combines centralization and distribution, and classifies the functions of the agent, allowing for a fair trade technology platform for all the parties in the electricity market auction. An economic grid resource scheduling model, proposed alongside a designed consumer utility function, is based on multi-agent collaborative technology and the market mechanics behind gambling[112]. Indeed, another model of task scheduling is based on permutation tree pruning[113], and considers two factors: execution cost and execution time of independent tasks. The issue of task scheduling is equationted as an n-level m-ary permutation tree. Traversing the permutation tree generates a scheduling scheme, and invalid path searching is avoided by using the pruning method.

3.2 Economic Theories can Provide a Solution to Solve Cloud Computing Resource Management Issues

The above survey indicates the prevalence of economic theory in projects and research. Economic driven cloud computing needs not only adapt to the fluctuations in resources in order to provide consistent, seamless, and transparent service in the cloud, but also be able to manage resources according to user needs and schedule tasks so that resource usage maximizes profits while consumers benefit from decreased consumption costs.

In the economy, the market is an autonomous decision-maker based on the distribution of the resource management mechanism, in which each market participant consumes based on market prices and personal preference. Managing cloud computing resources also requires a similar distribution of autonomous decision-making. Based on the computing resource are also a kind of society resource, the economy could be one of the efficient way to manage and allocate computing resources. Cloud computing resource transactions, however, are not independent of the economy, and thus cloud computing can also be redefined as an economic issue.

3.2.1 Cloud Computing as a Business-driven Technology

The Cloud computing continuously integrates internet related technology and infrastructure, meanwhile, in the real world, a lots of computing tasks in enterprises run in a distributed environment. Research and production in enterprises involve a large number of computing tasks. Presently, an increasing number of enterprises operate in a distributed computing environment. Due to the limitations of middleware all clients only have access to a static distribution of resources. Every single computation has to through the client / server mode to access limited resources. Some current Internet-based business models such as ASP（Application Service Provider）, SSP（Storage Service Provider）, IDC（Internet Data Centre）, have to access a static distribution of resources and are limited by the configuration[114].

However, IT environment and IT management are becoming increasingly complex as technology continues to respond to market pressures, and cloud computing technology can integrate and share existing computing resources to avoid blind expansion, reduce consumption costs, and increase computing capacity. The IBM "Blue Grid" has unified computing resources under the same management and control to improve computing power[115]without increasing costs.

3.2.2 Cloud Computing Technology Fit in with the Needs of the Society and Economic Laws

Cloud computing do distribution computing and a distribution commerce environment support a kind of technology. Relations of production must conform to the level of productive forces. Same reason, the clouding computing technique make the change of the commerce behaviour based on the distributed computing change. The computing is originate come from the apply demand, therefore the change of the apply demand will impact the technology of the cloud computing. The cloud computing presents the virtual resource as a service and use those kind of service to support all kinds of application system.

According to the law of conformity of production relations to the state of productive forces, all the economic activities that rely on cloud computing also followed the same principle. Currently, lots of countries and enterprises are investing money in cloud computing and related opportunities.

3.3 Using Economic Theory in Cloud Computing Resource Management

The framework and technicalities of cloud computing are closely related to economic theories, such as the market mechanism, the supply-demand relationship, the pricing mechanism, the optimized principle and so on. Even indirect economic thought may help solve many key issues in cloud computing.

3.3.1 The Law of Demand and Supply in Cloud Computing Environment

In economics, commodities can be divided into normal goods and low-grade goods. The demand for normal goods rises in tandem with consumer income level, while demand for low-grade goods is inversely related to consumer income level[116]. Demand for cloud computing services also increases with consumer income, classifying service as a normal commodity. Therefore, when all other factors are ignored, if the price of cloud computing services rises, then demand will decrease, and vice versa.

In a controlled environment, the quantity of the cloud computing service is the quantity of the cloud computing service available on the market.

Changes in price affect cloud computing service provision as follows: when cloud computing service prices rise then profit increases of providers. The provider will invest additional resources to increase the supply of the service. If profits do not increase, investment in cloud computing will decrease as investors switch to other, more profitable commodities or services, thus, reducing oversupply of cloud computing services. As a special commodity, cloud computing services thus satisfy the law of supply.

3.3.2 The Law of Diminishing Marginal Returns in the Cloud Computing Environment

The law of diminishing marginal returns in the cloud computing environment is as follows. In the cloud computing business model, if there is no investment, there will be no output, and so increases in investment lead to increases in output. In this context, the marginal returns are also increasing. However, when the input reaches a certain point within a certain range, output increases begin to shrink, although input and output still maintain a positive relationship. This deceleration is due to market saturation, during which increases in investment do entice new consumers but service output does not increase as sharply[117].

3.3.4 Monopolies in the Cloud Computing Environment

A monopoly is a market transaction. Buyers and sellers have differing degrees of control over the market price. There are three main types of monopolies: complete monopoly, oligopoly markets, and monopolistic competition[118]. An example of a complete monopoly would be if a government granted a provider sole rights to offer a service. An oligopoly would form if a cloud service provider held all key resources, such as super computing power, because they would then be the sole supplier of the resources. A monopolistic competitive market occurs when cloud service vendor produce similar products, but each vendor's products have different characteristics among their products. In this situation, vendors can freely access to the market, but they compete for the same customer groups; each vendor is not a price taker, but has a certain kind of monopoly power.

3.4 Summary

Economic issues are among the key problems of cloud computing. Economic laws and restrictions increasingly guide cloud computing, because cloud computing involves both resource allocation and a business model. The GESA, GRACE, and IBM commercial cloud, and a number of other commercial cloud models are representative of the development of the economics of the cloud. As market-driven technology advances, so too. Will the economics of cloud computing improve to better adapt to the market economy.

Chapter 4 Problem Identification

The main cloud computing resources could be used for providing computing ability, data storage and network capabilities. In order to meet the requirements, all the resource capabilities are flexible, cost-effective, have a high reliability, the distribution of applications and scalability features. The following problems have to face them first. Such as resource accounting, resource discovery, resource management model, cloud security, resource scheduling, and state records. Managing these aspects faces several problems, as will be outlined in this chapter[119].

4.1 Resource Accounting

Due to the pay as you go business model, users only pay for the services they consume, which correspond to the cost of providing the resource. Correspondingly, the cost of resource provision differs along the three types of cloud service: SaaS (Software as a Service), PaaS (Platform as a Service) IaaS (Infrastructure as a Service), and thusly so do accounting methods differ.

The main purpose of resource accounting is to build a foundation scenario for building banking transaction system. Different transactions between resource providers and consumers during the Cloud Bank life cycle need different resource pricing schemes and different accounting methods, both of which much be precise. For example, Amazon’s EC2 cloud computing system offers a pricing policy that bills according to rent time and resource used. Renting 4 CPUs, 10G memories and 100G hard disk at the same time for 1 hour will cost about $10. Pricing strategies differ by company, and there is one-sided pricing policy for the public. Lack of consistent market pricing combined with difficulty in measuring consumption challenge the ongoing development of the Cloud Bank system. Based on the Quality of service (QoS) guarantee and using original Cloud Bank Service-Level Agreement (CB-SLA), this thesis describes transactions of those related resources.

4.2 Resource Scheduling

The effectiveness of cloud computing resource management system and its acceptability are dependent on the implementation of a resource scheduling system. The dynamics and heterogeneity of the cloud computing determines the complexity of resource scheduling system. Cloud computing resource management scheduling system can be centralized, distributed and hierarchical. There are even econometric methods for scheduling.

In a centralized environment, a central scheduler organizes all of the resources using system information is gathered in the central machines. By comparison, in a distributed environment there is no central scheduler responsible for scheduling operations. A scheduler submits jobs to a remote system for distributed interaction, and a single component failure will not affect the whole cloud computing system. Fault tolerance and reliability are higher than the centralized system. However, a parallel program may be assigned to all parts of the resources in different domains, so different schedulers must execute tasks synchronously, which makes it difficult to optimize the scheduling system.

In the hierarchical scheduling model, there is a centralized scheduler. All jobs are submitted to the centralized scheduler, and each machine uses a separate resource scheduler for local scheduling. The main advantage of this structure is that different strategies can be used to for local and global scheduling.

Scheduling models based on econometric superiority are able to respond to supply and demand. In contrast to the other systems, the resources scheduling shifts from system-centric to user-centric, because the decision process is spread across all distributed users and resource owners.

Instead of using previous models, this thesis offers an original model based on banking systems. It uses the Pareto equilibrium to maximize benefits for all the participants, and changes scheduling from system-centric to user-centric. The thesis improved the 2×2 Pareto equilibrium to $n \times m$ participants to meet the needs of real cloud computing environment.

4.3 Cloud Computing Resource Transaction Risk Mitigation and Coping

Like all other information systems, stable and risk avoid are still the most important issue in cloud computing. The transaction safe is one of the most important issues of the cloud security. It can be divided into two areas: multiple information sharing enterprises, and individual users. The following are the most important information sharing security concerns:

- Safe equipment for cloud computing information systems should consider the stability, reliability, and availability of information systems equipment;
- The security of the cloud data centres, which also refers to data confidentiality, integrity, and availability;
- A secure environment for data use, which includes information use, confidentiality, integrity and conduct acts of controllability.

The resource model determines both the description of the resource system and the management of the cloud resource. Different descriptions of resources and service delivery methods depend on different resource management models. The organizational structure of the physical machine for cloud computing affects the mode of communication with the resources management system, and therefore determine the scalability of the integrated cloud computing architecture. The business and management model determines the service provision model and the business trading model. In the organizational structure of the centralized physical machine, the individual or a group controller schedules all the machines. This method does not work in cloud computing systems because of issues of scale. However, a distributed organizational structure can be imported onto cloud system machines. When combined with the unified trade model, then the cloud computing system provides high performance services for all the cloud computing types.

This thesis applies different strategy and methods based on the Cloud Bank model, according to the different period and features of the transaction to mitigation and predicted risk.

4.4 The QoS Issue

Applications require quantitative and qualitative characteristics to guarantee a level of service, of which the QoS is a reflection. In the network, QoS is a set of tools and standards. It gives network administrators control over bandwidth, delay, jitter and packet loss and other matters of comprehensive capacity. Usually, the network QoS only considers network issues, but in cloud computing systems, QoS should be extended the bandwidth of the network to the cloud computing nodes on processing power and storage capacity. Thus, in the cloud computing environment, resource management systems should also be able to provide a set of end to end QoS, and ensure the computing capabilities of each node in the distributed environment.

This thesis offers a new CB-SLA to describe and monitor transactions and guarantee the service level: the newly proposed and designed XML schema. All contracts are signed based on the CB-SLA, thus guaranteeing QoS for all involved.

Chapter 5 Research Approaches to Banking Models for Cloud Computing Resource Management

Resource management and the scheduling are the key issues in the cloud. As a part of the cloud environment, both resource providers and consumers have their own requirements and strategies. The consumer needs the resources to process their task. At the same time, the resource provider has idle resources that can be provided. The market theory based cloud computation model tries to follow the principle of supply and demand in the market transaction, and thus facilitate trades that maximize mutual benefit.

From the view of computing resources and users, there are three forms of the cloud organizations: public cloud, private cloud and hybrid cloud. The following figure demonstrates the three kinds of clouds.

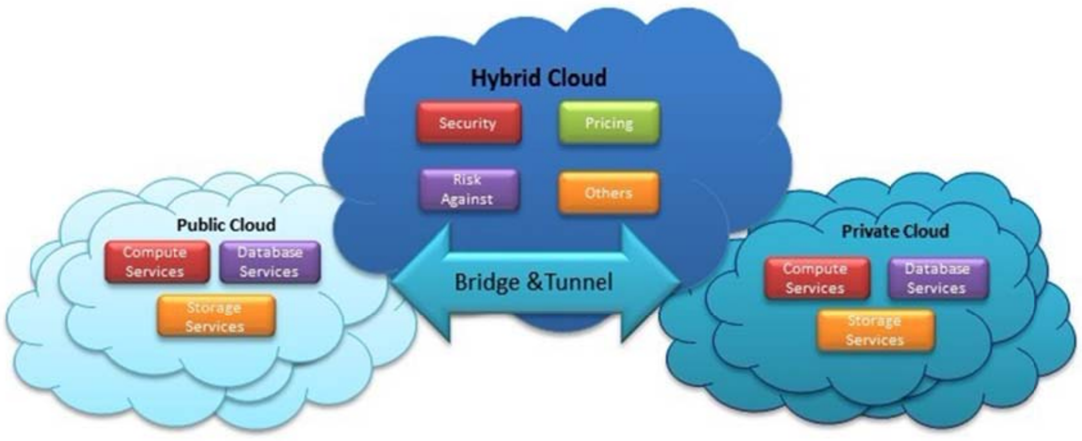


Figure 5. 1 The Three Kinds of Cloud Organizations

A Hybrid cloud exists when the cloud infrastructure comprises two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds)[120]. For security reasons, not all the enterprises put their information into the cloud. Most of them will store data individually and try to use the application through the public cloud and others cloud resources. That is not to say the private clouds and public clouds insolate each other, but rather that they collaborate with each other. The typical hybrid cloud user uses private clouds for storage, handling databases and services (to eliminate the need to buy additional hardware), and takes advantage of the peak demand period to complete public cloud data processing needs. Currently, there are already many companies that are moving in the direction of developing the architecture for such a concentrated cloud (cloud-bursting). This is also the key to maximizing benefits.

Therefore, having surveyed some of existing economic models, there is no one model alone that can support resource transactions that allow all parties to know the result beforehand. Inspired by the above works, this thesis proposes an economic-based cloud resource allocation-scheduling algorithm. The new mechanism is limited by the parameters of the economic model that can deliver great value in the cloud resource allocation. The aim of this mechanism is to maximize the biggest value for the three parties in this cloud computing transaction while maintaining acceptable QoS. This thesis equation the integrated design of scheduling into a constrained optimization problem.

5.1 Banking Model

After investigating existing banks in the real world, we found that the purpose of banks is to accept deposits from the public and allocate the sum to offer services to all kinds of users. The “money” in the bank is actually in different forms. There are different currencies, mortgages, or even gold. After banking operations, all the “money” has minimal monetary power and has been reorganized as all kinds of products to service different users.

Inspired by commercial bank’s deposit-loan transaction, this thesis advance Cloud Bank as shown in figure 5.2. This thesis has a Cloud Bank definition as follows:

The Cloud Bank is: according to the mode of operation of commercial banks and the laws of the market, the different roles involved in cloud computing transactions, respectively, defined as cloud resources provider, service consumers and the Cloud Bank. Its infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. The Cloud Bank is a hybrid cloud model that uses a new resource management transaction structure driven by the microeconomics, and banking theory. The hybrid cloud range is further extended so that it can provide users with more comprehensive, stable, and high-quality services.

In this model, the whole cloud computing environment is classified into three roles: cloud resource providers (CRP), the Cloud Bank, and cloud resource consumers (CRS). The resource provider is equivalent to the depositors at commercial banks, and resource consumers are similar to borrowers, and the Cloud Bank plays the role of commercial banks in deposit-loan transaction. The Cloud Bank pays the depositors certain interest according to the storage time, and charges a loan fee according to the interest and duration of executed tasks. We introduce the optimal deposit-loan ratio theory of commercial banks to guide Cloud Banks' daily managing to do the following: encourage resource providers to deposit their resources into the Cloud Bank and incentivize resource users to borrow resources from the bank; realize a win-win situation for both supply and demand sides; balance supply and demand; and maximize the Cloud Banks operator's profit.

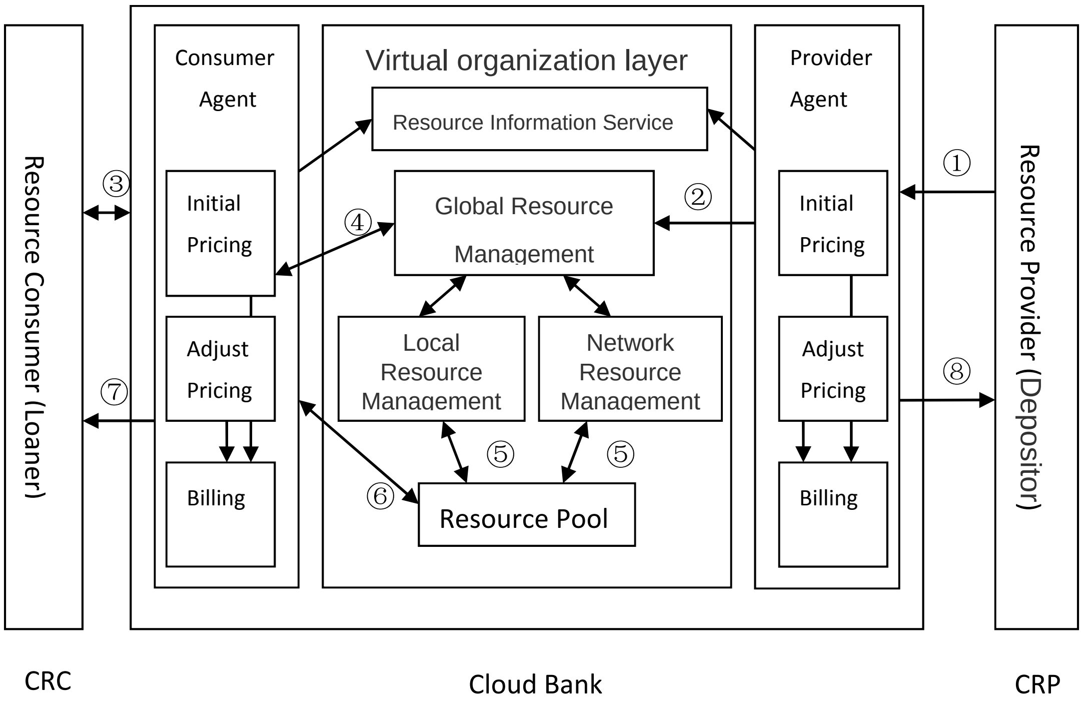

Resource Consumer (Loaner)
Consumer Agent
Initial Pricing
Adjust Pricing
Billing
Virtual organization layer
Resource Information Service
Global Resource Management
Local Resource Management
Network Resource Management
Resource Pool
Provider Agent
Initial Pricing
Adjust Pricing
Billing
Resource Provider (Depositor)
①
②
③
④
⑤
⑤
⑥
⑦
⑧
CRC
Cloud Bank
CRP

Figure 5. 2 The Architecture of Banking Based Cloud Resource Allocation (BCRA)

Based on the Cloud Bank models and the deposit and loan process for resources, the operating process of Cloud Banks are:

Step 1: Depositors store their resources in the bank and state storage time. Cloud banks calculate the initial price of storage through the provider agent, and display deposit interest rates;

Step 2: Provider agent registers resources to global resource manager, and indicates the storage time and rates;

Step 3: Consumers apply to the bank for resources according to predicted time requirements and budget.

Step 4: According to consumer's proposed task execution time and budget, the consumer agent uses the current loan interest rate to calculate whether there are sufficient available resources, and provides the outcome to the consumer. The consumer weighs the options, and decides whether or not to rent.

Step 5: If the consumer decides to rent the resources, then he\she sends the changed task execution time and budget to the consumer agent, who applies for resources from global resource manager, otherwise the consumer quits the Cloud Bank platform.

Step 6: The global resource manager finds if there are available resources first through local resource managers. If the former is not possible, then local and network resource managers collaboratively allocate resources. They return the required resources, packed into resource service set, to the consumer agent.

Step 7: The consumer agent sends the resource service set to the consumer and charges fees according to initial price and loan interest rate. If task execution failed or was not completed on time, then the Cloud Bank compensates the consumer's loss and completes this transaction.

Step 8: When resources stored in the Cloud Bank expire, the provider agent calculates fees according to storage time, initial resource price and deposit interest rate, and pay the depositors this fee.

5.2 How Does the Cloud Computing Follow the Real Bank to Do Transaction

As a cloud computing resource mechanism, the system model makes use of the economic model to manage cloud resources. This thesis assumes that the behaviour of cloud resource consumers is predictable and provider resources are dynamic. Therefore, we reorganized all possible individually idle resources into the bank to provide them as a resource pool to the consumer. The basic idea is to mark off all resources according to the QoS level and resource type. All transactions are monitor and audited by the third party—the bank.

The following figure shows the relationship among the Cloud Resource Consumer (CRC), Cloud Resource Provider (CRP) and Cloud Bank (CB)

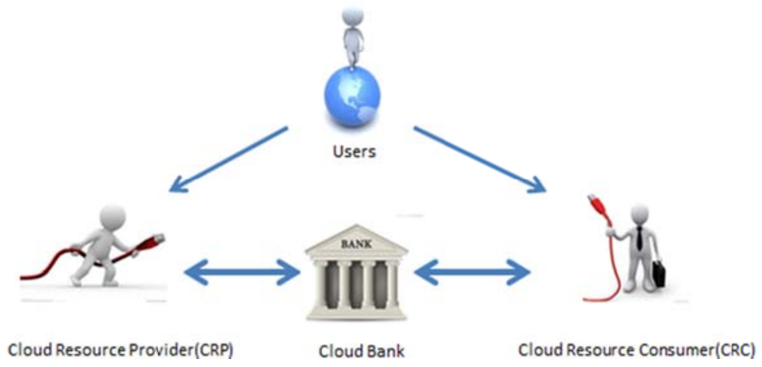
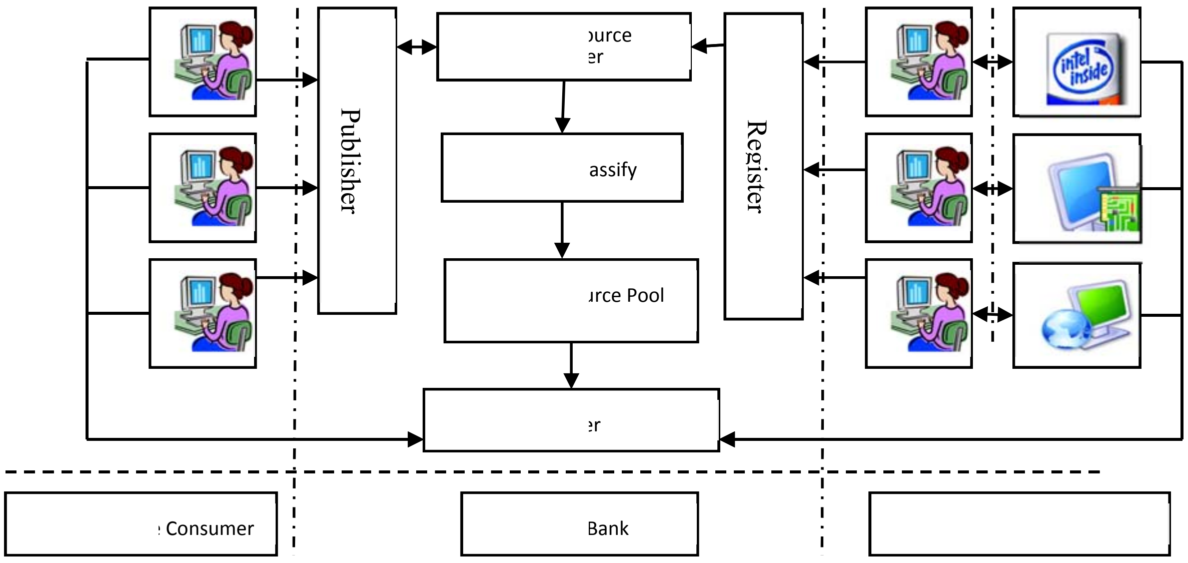


Figure 5. 3 Roles in the Cloud Bank

This figure focuses on the users and the bank. The users can come from any physical location and the Cloud Bank is a virtual resource Agent. The users (CRP, CRC) are “Global”. They could be an individual person or an organization. The Cloud Bank tries using banking mechanisms to integrate all related parties together and maximize participant profit.

The following figure demonstrates how does the banking theory is used to manage the IaaS based cloud resource allocation.



- There are three parts composed the architecture of BCRA
- Cloud Resource Consumer (CRC): The service consumers do not need to talk to the

resource providers. They get all available resource information through the publisher of the bank. Then, CRC can employ the resources to do their job. Consumer agent for their charges them.

- Bank: As a transaction coordinator, it stores all the resource provider information into the Cloud Resource Manager (CRM), inside of the bank. The CRM sends information to the Resource Classify Centre (RCC), which separates resource classes into a virtual pool. The Virtual Resource Pool (VRP) stores (already registered and marked as available) all the available resources. The bank not only links resource providers and service consumers, but also masks relations between them. One of the principles of banking transactions is symmetrical information sharing between the depositor and lender. In the cloud environment, the symmetrical information is a QoS based transaction.
- Cloud Resource Provider (CRP): The CRP registers their idle/leftover resources in the Register Centre of the bank. They open an account with the bank with all the necessary parameters. They still own the resources via the policy, but, if some of the CRCs require them, the CRP has to accommodate the CRC first according to the banks regulations.

5.2.1 Optimal Deposit-loan Ratio Theory in Cloud Banks

Deposit-loan ratio [122] of Cloud Bank refers to the proportion of total loans to total deposits. As mentioned above, this thesis compares resources stored in the Cloud Bank to deposits. Corresponding borrowed resources are compared to loans. This thesis assumes that the more resources on loan, the better. But economics research has found that transforming deposits into loans is not the more the better. With the increase of income of banks, the risk is also increased. Some banks will result in serious risk of bankruptcy.

When resources are in use, various faults and mistakes are often frequent and uncontrolled. However, new resources must replace them, so Cloud Banks must reserve spare resources to replace failed resources to ensure timely task completion. Otherwise, Cloud Banks will suffer loan default loss. Thus, the deposit to loans conversion ratio is not the more the better, but rather there exists an optimal deposit-loan ratio. Under the premise of this ratio it is possible to maximize benefits for all participants. Building from this premise, this thesis expects to find this ratio in our management model. We adjust deposit and loans' interest rate through this ratio to maximize benefits for participants and optimize cloud services.

5.2.2 Identifying Factors Affecting the Cloud Bank

There are four factors that affect the Cloud Banks' operating costs[123]:

(1) Cloud Bank cost losses caused by owners who withdrew their resources that are rented out by Cloud Banks and are being used by users. In short, savers default cost.

(2) The cost of compensating consumers whose tasks were not complete on time according to the contract, in order to maintain the Cloud Banks' reputation. In short, loan default cost.

(3) The cost of monitoring rented out resources, namely loans supervision cost.

(4) The cost of daily maintenance on the Cloud Bank's operating system, namely the daily operation cost.

5.3 The Pricing Schema for Cloud Computing

The price adjustment algorithm of resource is divided into two categories in the economics computing area:

1. Centralized synchronous algorithm for price adjustment.

The algorithm uses the idea of total supply and demand balance, according to all the resources in the total supply and demand, to simultaneously adjust the prices for all resources.

2. Distributed price adjustment algorithm.

Distributed price adjustment algorithms classify the resource first, and then based on supply and demand for each type of resource adjust the price in order to balance supply and demand.

5.4 Avoiding Banking Risk in the Transaction

The new Basel capital accord offers algorithms to judge the level of resource. There are four elements involved in computing risk for the bank probability of default, loss given default, exposure at default, and maturity[123]. This model emphasizes restricting the provider. In addition, the probability of default model for credit risk is used to forecast the safety of the resource supply. The logistic regression model issued to control some resource factors.

5.4.1 Logistic regression model

The Logistic Regression model is one of the generalized linear models. The idea of Logistic Regression Model is to separate the providers or consumers who choose to renew their contract from those who make the different decision, and then study the two groups and find out the common features. According to the mapping between these common features and one provider's or consumer's own features, the probability can be worked out.

The basic equation for Logistic Regression Model is

$$\text{logit}\big(Pr(Y = 1|X)\big) = \beta_0 + \beta_1 * X_1 + \cdots + \beta_m * X_m \tag{1}$$

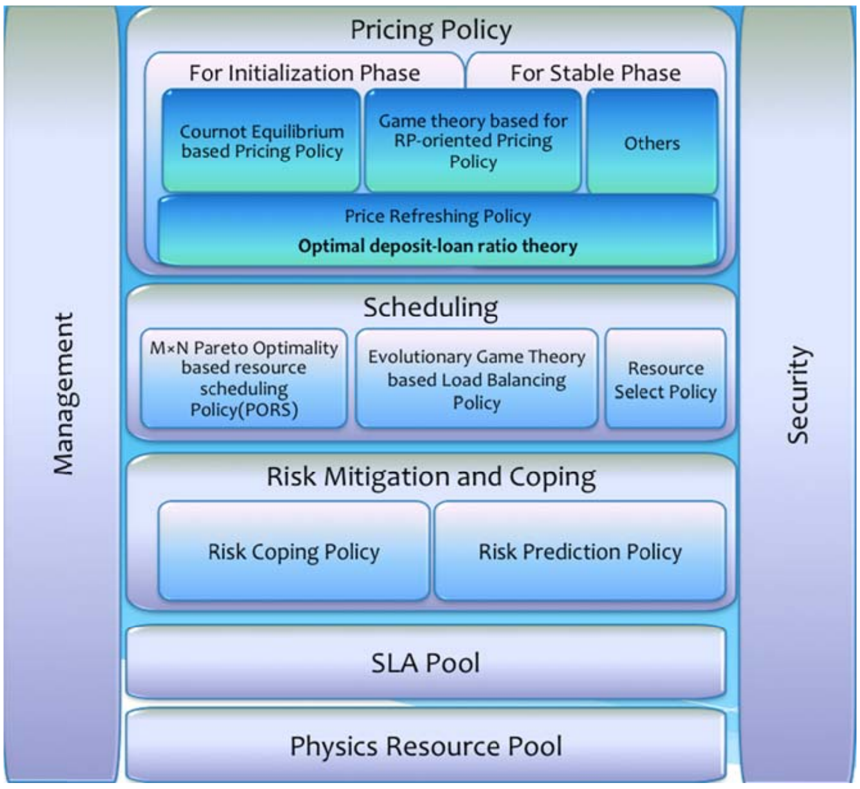
In (1), $\text{logit}(x) = \ln(x/(1 - x))$; $\mathrm{Y} = 1$ means choosing not to renew the contract.$X$is the set of the features of a provider or a consumer. $Pr(Y = 1|X)$ is the probability of choosing not to renew the contract under the condition X. $\{X_i\}$ is the set of all the common features. $\{\beta_i\}$ is the set of the model parameters; each β_i can reflect the importance of the X_i to the prediction[124].

5.5 Cloud Bank Scheduling and the Pareto Optimality

Pareto optimal state is generally considered to be an optimal allocation of social resources, or the effectiveness's standard of resources allocation. The Pareto optimal state is when the reallocation of resources to make some people better without reducing the situation of a person or the situation of some people. In this state, if we do not damage a person or persons' benefit, we cannot make another person or another persons' situation better. In other words, if a person or some people's situation is not changed, then the situation of some another people are best. If we want them to be better, then the situation of some people must worsen. In the welfare economics, this thesis discusses the Pareto Optimal mainly from the consumption and production. This maps the relationship of the resource providers and resource users in cloud resource management. In the Cloud Bank, the result of resource scheduling among tasks is to reach the Pareto Optimality and make the system run smoothly.

5.6 Interior Components of the Cloud Bank

Based on the above discussions, the thesis proposes logical interior components for Cloud Bank architecture. The following figure provides more details about each component, and they the relationships among them.



Pricing Policy
For Initialization Phase
For Stable Phase
Cournot Equilibrium based Pricing Policy
Game theory based for RP-oriented Pricing Policy
Others
Price Refreshing Policy
Optimal deposit-loan ratio theory
Scheduling
M×N Pareto Optimality based resource scheduling Policy(PORS)
Evolutionary Game Theory based Load Balancing Policy
Resource Select Policy
Risk Mitigation and Coping
Risk Coping Policy
Risk Prediction Policy
SLA Pool
Physics Resource Pool
Management
Security

Figure 5. 5 The Interior Components of the Cloud Bank

There source trading model is divided into the following five levels: physical resource Pool, SLA pool, risk mitigation, scheduling and pricing policy.

- The pricing policy includes two parts:

  For initialization phase: Cournot Equilibrium based pricing policy

  For stable phase: Stackelberg model based pricing policy

  For centralized synchronous updating: Deposit-loan ratio

- Scheduling:

  $M \times N$ Pareto Optimality based resource scheduling policy (PORS)

  Evolutionary game theory based load balance policy (in the future)

  Resources select policy

- The risk mitigation and coping include two parties:

Risk coping policy

Risk predication policy

- SLA pool:

  It is not a real physical level, in the logical, all the resource metadata and transaction information have to base the discretion of the SLA. It will give all the necessary parameters and elements in the related entities.

- Physics resource pool:

  Real distributed resources include the CPU, memory, hard disk and network environment. They are autonomous and join the Cloud Bank

5.7 Summary

This chapter proposes a commercial bank based hybrid IaaS cloud computing model. The model attempts to simulate the relationship among the real commercial banks' depositors, lenders and banks. It tries to set up a new hybrid cloud computing model for the resource trading. The Cloud Bank model divided users as resource providers and resource consumers. The resource providers can get the profit from the transaction in the hybrid Cloud Bank like a real bank. The whole resource trading model is divided into the five levels: Physical Resource Pool, SLA Pool, Risk Mitigation, Scheduling and Pricing Policy. Therefore, this model has a different pricing policy for them, which are resource provide price, and resource consume price. It also tries to follow the way of risk-mitigation strategies from the commercial bank to set up a risk coping and mechanism for hybrid cloud resource trading. The following chapters will discuss the risk coping, pricing and scheduling related content in details.

Chapter 6 Research Approaches for Risk Mitigation and Coping

Cloud computing environments are as dynamic as they are distributed. Resource providers are distributed physically and the provided resources are heterogeneous, creating many uncertainties that could affect the reliability of the Cloud Bank. For example, some resources might be unavailable due to something as simple as a broken cable. Small though they might be, uncertainties of this kind could potentially cause some risks that could seriously affect both the reliability and reputation of the Cloud Bank. Accordingly, assessing the risk mitigation and management is a very important part in the whole Cloud Bank model.

The Cloud Bank model is essentially a resource management model based on already established economic principles that have been distilled and used to rebuild a modern digital system. Initially inspired from the commercial banking model, the Cloud Bank has some uncertainties rather similar to commercial banks. For example, the depositors in the commercial bank may withdraw their money at any time. Though in the cloud these are not always tangible assets such as hard currency, the situation is quite similar in that the resources of the Cloud Bank may be unavailable in a particular quantity and at a particular time. To then coping the Cloud Bank's risks adequately, it is worth examining how commercial banks manage their risks.

6.1 The Risk Mitigation and Management in Commercial Banks

The risks contained in the bank's principal activities are not all caused by the bank. Some risks may be caused by the depositors or the borrowers. For different reasons, the risks can be divided into six types: systematic or market risk, credit risk, counterparty risk, liquidity risk, operational risk, and legal risk. In many instances the institution will eliminate or mitigate the financial risk associated with a transaction by proper business practices. In others, it will shift the risk to other parties through a combination of pricing and product design[126].

Risks facing all financial institutions can be segmented into three separable types, from a management perspective. These are:

1. Risks can be eliminated or avoided by simple business practices.
2. Risks can be transferred to other participants.

3. Risks must be actively managed at the firm level.

In the first of these cases, the practice of risk avoidance involves actions to reduce the chances of idiosyncratic losses from standard banking activity by eliminating risks that are superfluous to the institution's business purpose. Common risk-avoidance practices here include at least three types of actions. The standardization of process, contracts, and procedures to prevent inefficient or incorrect financial decisions is the first of these. There are also some risks that can be eliminated, or at least substantially reduced through the technique of risk transfer. Markets exist for many of the risks borne by the banking firm. Interest rate products such as swaps or other derivatives can transfer interest rate risk. Borrowing terms can be altered to effect a change in their duration. Finally, the bank can buy or sell financial claims to diversify or concentrate the risks that result from servicing its client base. There are still some risks should be absorbed at the bank level. These risks should be monitored and managed efficiently by the institution[126].

In light of the above, what are the necessary procedures that must be in place in order to carry out adequate risk coping? In essence, what techniques are employed to both limit and manage the different types of risk, and how are they implemented in each area of risk control? It is to these questions that we now turn. After reviewing the procedures employed by leading firms, an approach emerges from an examination of large-scale risk coping systems. The management of the banking firm relies on a sequence of steps to implement a risk coping system. These can be seen as containing the following four parts[126]:

1. Standards and reports.
2. Position limits or rules.
3. Investment guidelines or strategies.
4. Incentive contracts and compensation.

6.2 The Risk Mitigation in the Cloud Bank

Although the Cloud Bank model has some uncertainties that are similar to the commercial bank models, the Cloud Bank has its own features that are very different from the commercial bank. For example, the borrowers always cause the credit risk in commercial bank, but in the Cloud Bank, the consumers cannot cause risks like credit risk, because the Cloud Bank never lends out the real resources to the consumers. The real resources are still in the hands of the resource providers.

There are two features about the resource transactions in Cloud Bank that should be emphasized: unchanged ownership of the resources and the consumer's unclear resource requirements.

As mentioned above, the Cloud Bank model is an open public cloud computing model. The resources and ownership are physically distributed on the Internet[127]. The Cloud Bank virtualizes these heterogeneous resources and provides homogeneous services to the consumers, but the ownership of the resources has not changed during the transaction. This is the biggest difference between the commercial bank and the Cloud Bank. Because of the unchanged ownership, the quality of the services that the Cloud Bank could provide almost depends on the quality of the resources provided by the providers.

Real cloud computing applications are very complex and commercial. Even the experts in this area cannot give a clear and precise resource requirement about an application, so it is possible that a consumer wants to change the resource requirements when the contract expires. The change of the resource requirements might cause risk, such as when a consumer wants to renew the contract, but the resources should be given back to the providers. In order not to aggrieve the consumer's benefits, the Cloud Bank could only renew the contract. Then, because of the unchanged ownership of the resources, the Cloud Bank should search for other resources to provide to the consumer. If the search result failed, the credit of the Cloud Bank would be affected.

There is one significant feature about the damage of risk in Cloud Bank that should be emphasized: fast and irreparable damage without any cushioning. As mentioned above, cloud computing applications are commercial, so keeping the applications running continuously is very important. If a risk arises and an application stops running, the damage will be caused immediately and irreparably. There are no equivalents in Cloud Banking to mortgages in commercial banks, which provide some cushioning when risks arise. Therefore, preventing risk from arising is the most important part in the strategy of the risk mitigation in the Cloud Bank. Predicting risks for early mitigation is the best and the most important way.

Based on features for the Cloud Bank, the strategy of risk mitigation in Cloud Bank is divided into two parts: prediction and coping.

6.2.1 The Classification of Risks in Cloud Bank

Before identifying how to predict and manage risks in Cloud Bank, we should first establish what kinds of risks are inherent in the Cloud Bank. After researching the features of Cloud Bank, risks can be divided into the following categories: credit risk, liquidity risk, operational risk, and other risks and so on. The table 6.1 shows the relationships between the risks and the roles in the Cloud Bank.

Table6. 1Cloud Bank Risks Classification

	Providers	Cloud Bank	Consumers
Credit Risk	Caused by		
Liquidity Risk		Caused by	Caused by
Operational Risk		Caused by	
Other Risk	Caused by	Caused by	Caused by

For the features of Cloud Bank reason, the credit risk never cause by the consumer, because the consumer do not really take any resource physically. Credit risk arises from non-performance by a resource provider. The environment of cloud computing is dynamic, so a failure by the providers to provide resources during the contract at any time is an activator passive behaviour that might cause the loss of the resources, which would affect the service provision, and the consumer, finally resulting in a loss of the consumers' benefits.

Liquidity risk arises from the Cloud Bank's inability to provide the services because resource providers withdraw too many resources. It can be described as follows: when the contract expires, the resource providers take their resources back, but these resources may still be rented out to the consumers. If the resource pool does not have enough resources to replace these removed resources, then the quality of service of the Cloud Bank would inevitably decrease. The benefits of the consumers would be aggrieved.

The Cloud Bank's operational risk happens when the Cloud Bank's internal operational management and control mechanisms fail. Other risks mean those risks might have bad effect on the Cloud Bank. For the reason of topic of operational risk and other risk are too big to management, this thesis will conduct research in the future. The paper just focuses on the credit risk and liquidity risk.

6.2.2 The Strategy of Risk Mitigation

In commercial banks, if a risk arises there are some remedial measures to reduce the damage. For example, when credit risk arises in a commercial bank (i.e. the borrower cannot repay the money) mortgages can be used to repay the cost. These mortgages are cushions that reduce the risk damage. In Cloud Banks, however, there are no such things as mortgages that can help reduce the risk damage. The providers cannot to ask to offer a backup of their resources as a strategy to protect mitigation credit risk. All cloud computing applications run on the cloud computing platform, so if one risk arises, the damage occurs immediately, and a domino effect follows much more quickly in the cloud computing environment than in the commercial environment. There is not enough time to start remedial measures after the risks arise. Because of the features of risk damage in Cloud Bank, the strategy of risk prediction is a very important part in the risk mitigation.

6.2.2.1 The Strategy of the Credit Risk Prediction

Should risks to the Cloud Bank’s credit arise during the contract, the appearance is the resource loss. Macroscopically speaking, there are two kinds of resource default in Cloud Bank model: the first type is defined to be the subjective reason-caused default, and the second type is defined to be the objective reason-caused default.

1. Subjective Factors-caused Default

Subjective factors-caused default is such a kind of default which is artificially induced by the resource entity holders (CRP).

The subjective factors-caused default during the contract, or artificial-reasons default, means that the resource providers sign out from the Cloud Resource Agent (CRA) on their own initiative (e.g., the resource provider shut down the agent). The CRP must sign a contract with CB before the providers’ resource joins into the resource pool, and the contract contains the penalty provisions. Consequently, considering the penalty cost and reduced credit rating, the cost of default is quite expensive for CRPs. So, the resource provider`s most probably default motivation is the goal of chasing a higher earnings, after the appearance of a higher resource purchase price by another Cloud Bank in the market.

Based on the analysing above, the default profit value can well format the subjective factors-caused default reason.

2. Objective Factors-caused Default

Different from objective factors-caused default, objective factors-caused default is commonly induced by machine fault or Internet failure.

Objective factors-caused default is mostly caused by random incidents such as Internet failure, power failure, or computer virus outbreaks. These incidents induce a large number of resource losses in a short time and even cause service quality shock. Certain indicators such as network delay, resource working historical credit record and response delay can reflect the resource real-time situation with statistical significance. The Cloud Bank compiles these variables into the a standardized Quality of Resource (QoR) value in form of Service-Level Agreement(SLA) according to their practical needs, and this value can measure objective factors-caused default possibility.

After the analysis above, this thesis will deliver a cloud resource default ratio (DR) prediction method on basis of the multiple linear regression analysis.

Multivariate statistical analysis is a branch in mathematical statistics, the main purpose of which is to research how to efficiently process and analyse data restricted by random factors, and set out predictions and inferences for the issue under investigation. Linear regression is an approach to modelling the relationship between a dependent variable y and one or more independent variables X. When the numbers independent of variables are more than one, it belongs to multiple linear regression analysis. A multiple linear regression analysis is carried out to predict the values of a dependent variable Y, given a set of p explanatory variables (x1, x2,....,xp). In these notes, the necessary theory for multiple linear regressions is presented and examples of regression analysis with census data are given to illustrate this theory[128].

Normally, the basic procedures of a multiple linear regression analysis can be outlined as follows and also shows as figure 6.1.

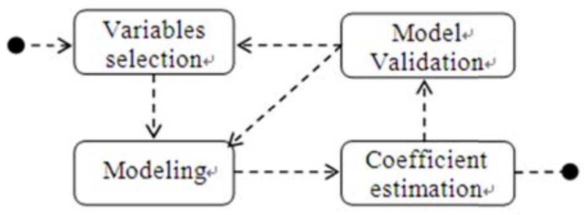


Figure 6. 1 Multiple Linear Regression Analysis Procedure

1. Explanatory variables selection:

The model makes the prediction according to some significant explanatory variables (independent variables). The guiding principle of the explanatory variables selection can be explained as follows:

The explanatory variables must have significant effects on the dependent variable, and they must have closely linear correlation relationship.

The relationship between independent variable and dependent variable must have actual significance.

There should be exclusiveness to a certain degree between explanatory variables;

The explanatory variables should have complete historical data.

2. Modelling:

The model equation will be in such form:

$$\begin{cases} y = b_0 + b_1 x_1 + b_0 x_1 + b_0 x_1 + \cdots + b_k x_k + e \\ \hat{y} = b_0 + b_1 x_1 + b_0 x_1 + b_0 x_1 + \cdots + b_k x_k \end{cases} \tag{1}$$

In the equation above, $x_1 \cdots x_k$ are all explanatory variables. y is the dependent variable. Particularly, $\hat{y}$ is the predictive value. And e is the error between $\hat{y}$ and y. $b_0 \cdots b_k$ are called regression coefficients.

Regression coefficient estimation:

Each historical data period (including y_t and $(x_i)_t$) is substituted into the second equation, after which the regression coefficients are determined by solving multiple linear equations.

After regression coefficients have been obtained, we insert the current data ($(x_i)_{t+1}$) into the equation to obtain the prediction value $\hat{y}$.

3. Model validation and correction

After the multiple linear regression analysis prediction, it is important to validate and correct the model. Commonly used methods include regression coefficients fitting degree testing (T testing) and regression equation fitting degree testing (F testing).

Now we will use the multiple linear regression analysis method to build the prediction model according to the steps outlined above.

As the reasons leading to resource default analysed in table 6.1 can be classified into objective factors-caused default and subjective factors-caused default, and the prediction model contains at least two explanatory variables (binary regression). The explanatory variables are formatted into as follows：

① Default Profit (M):

Default profit value is formalized subjective factors-caused default variable. Intuitively, its meaning is the benefit a CRP can get if CRP decides to default.

The computational equation is as follows:

$$M = (I_{\max} - I) - B \tag{2}$$

I is the resource purchase price by the Cloud Bank the CRP in current contract, and $I_{\max}$ is the highest among all Cloud Banks in the market. So, $(I_{\max} - I)$ is the resource price discrepancy B is the default penalty specified in the contact.

② Quality of Resource (QoR) Difference Value (R):

The computational equation is as follows:

$$R = \begin{cases} \tilde{Q} \\ \tilde{Q} - \frac{1}{m}\sum_{j=1}^{m} Q_{\mathrm{j}} \\ 0 \end{cases} \tag{3}$$

Q_{j} is the resource quality evaluation value. $\tilde{Q}$ is the resource quality benchmark value set by Cloud Bank, which is set to be the ideal resource quality. The value`s setting reflects the resource quality requirement of one Cloud Bank. Because Cloud Banks must prevent a large amount of resource loss in a shout time, a variable is needed to represent the average level of the resource quality in the whole visualization resource pool. $\frac{1}{m}\sum_{j=1}^{m} Q_j$ is the average resource quality value, which reflects the general state.

Both of the two variables are positively correlated with the default ratio.

After obtaining explanatory variables, we can get the formalized possibility of default model based on binary linear regression.

The regression equation is as follows:

$$\hat{P}_t = b_0 + b_1 C_t + b_2 R_t \tag{4}$$

$\hat{P}_t$ is DR(default rate) predictive value of the $t - t_h$ observation period. C_t and R_t are default cost value, and QoR difference value in the $t - t_h$ observation period, which are the explanatory variables of regression equation. b_0, b_1 and b_2 are regression parameters.

Regression parameters (b_0, b_1 and b_2) can be solved by the following linear equations:

$$\begin{cases} \sum_{t=1}^{n} P_t = nb_0 + b_1 \sum_{t=1}^{n} M_t + b_2 \sum_{t=1}^{n} R_t \\ \sum_{t=1}^{n} M_t P_t = b_0 \sum_{t=1}^{n} M_t + b_1 \sum_{t=1}^{n} M_t^2 + b_2 \sum_{t=1}^{n} M_t R_t \\ \sum_{t=1}^{n} R_t P_t = b_0 \sum_{t=1}^{n} R_t + b_1 \sum_{t=1}^{n} M_t R_t + b_2 \sum_{t=1}^{n} R_t^2 \end{cases} \tag{5}$$

And the matrix form is:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} n & \sum_{t=1}^{n} M_t & \sum_{t=1}^{n} R_t \\ \sum_{t=1}^{n} M_t & \sum_{t=1}^{n} M_t^2 & \sum_{t=1}^{n} M_t R_t \\ \sum_{t=1}^{n} R_t & \sum_{t=1}^{n} M_t R_t & \sum_{t=1}^{n} R_t^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_{t=1}^{n} P_t \\ \sum_{t=1}^{n} M_t P_t \\ \sum_{t=1}^{n} R_t P_t \end{bmatrix} \tag{6}$$

The Cloud Bank should accumulate historical data over a certain time frame and compile it into a historic data table. The historical data is required to predict the $(t+1) - t_h$ period. Every M_t and R_t in the historic data table are substituted into the equation (5), after which we can obtain the regression coefficients: b_0, b_1 and b_2. Then substitute the regression coefficients and $(t+1)-t_h$ explanatory variables (M_{t+1} and R_{t+1}) into the equations (4), to obtain the predictive value.

With the system's developing, the default reason may have some changes, and some explanatory variables may no longer have any significance to the model at the same time that new decisive factors may occur. So it needs to verify the correctness after the event it been predicted is over, and make necessary correction to adjust the system developing.

There are some classic testing methods that can be used to test the explanatory variables and the whole model. One is a regression variable fitting degree testing (T-testing), and the other is the regression equation fitting degree testing (F-testing).

Regression variables fitting degree testing (T-testing):

The purpose of regression variables fitting degree testing is to verify whether the selection of each regression variable is correct. Consequently, the explanatory variable with a significant effect can be reserved; otherwise it will be displaced by another one.

The testing equation as follows：

$$t_i = \frac{b_i}{S_P\sqrt{E_{ij}}} \tag{7}$$

t_i is the value of the statistic b_i is regression coefficients. S_p is the standard error estimation value. E_{ij} is the $(j-t_h)$ element of the main diagonal of the inverse matrix which is matrix to solving the regression coefficients in the multiple linear regression equation. In terms of binary linear regression, the E_{11} and E_{22} can be generated from such equation as below：

$$\begin{cases} E_{11} = \dfrac{H_{22}}{H_{11}H_{22} - H_{12}^2} \\ E_{22} = \dfrac{H_{11}}{H_{11}H_{22} - H_{12}^2} \end{cases} \tag{8}$$

And H_{11}, H_{22}, H_{12} are:

$$\begin{cases} H_{11} = \sum_{t=1}^{n}(M_t - \overline{M}_t)^2 \\ H_{22} = \sum_{t=1}^{n}(R_t - \overline{R}_t)^2 \\ H_{12} = \sum_{t=1}^{n}(M_t - \overline{M}_t)(R_t - \overline{R}_t) \end{cases} \tag{9}$$

S_P is the standard error estimation value, and can be solved from following equation:

$$S_P = \sqrt{\frac{\sum_{t=1}^{n}(P_t - \hat{P}_t)^2}{n-k-1}} \quad (k=2) \tag{10}$$

k is the number of independent variables in a multiple linear regression equation.

Regression equation fitting degree testing (F-testing)

The purpose of regression equation fitting degree testing is to verify the significance of the whole regression equation. All explanatory variables are closely related to the dependent variables.

The testing equations as follows:

$$F = \frac{\sum_{t=1}^{n}(\hat{P}_t - \overline{P}_t)^2 \Big/ k}{\sum_{t=1}^{n}(P_t - \hat{P}_t)^2 \Big/ n-k-1} \quad (k=2) \tag{11}$$

According to the given significance level α (e.g. 0.95) and the degree of freedom (2, n-3), it can obtain the critical value F_a by F-distribution table. If the $F > F_a$, the regression equation has significant, and regression effect is obviously.

The whole progress of resource default risk prediction and response method is listed as follows:

Risk Prediction Stage:

Before the $(t+1) - t_h$ period`s start, substitute the historical data (M_t and R_t) into the equation (5), then the regression parameters can be solved; then substitute the current M_{t+1}' and R_{t+1}' into the equation (4), the DR prediction value ($\hat{P}_{t+1}'$) can be generate.

Risk Response Stage:

$\hat{P}_{t+1}'$ is the important parameter to activate the risk response mechanism. The $\hat{P}_{t+1}'$ means the severity of resource default. If the M_{t+1}' is the significant influencing factor, the Cloud Bank should preferred modify softness response strategy (pricing strategy); else if the R_{t+1}' is the significant influencing factor, the Cloud Bank should activate the hardness response strategy (start reserved resources).

Model Validation and Correction Stage:

Firstly, substitute the adjusted M_{t+1} and R_{t+1} into the equation (4) again, then $\hat{P}_{t+1}$ (the predictive value after model correction) can be solved. When the $(t+1)-th$ period is finished, the real default ratio value P_{t+1} will be received. The M_{t+1}, R_{t+1} and P_{t+1} should be recorded to the historic data table. Then, test the fitting degree between $\hat{P}_{t+1}$ and P_{t+1} according F-testing, and make the model correction.

6.2.2.2 The Strategy of the Liquidity Risk Prediction

The Cloud Bank's liquidity risk may arise when the contract expires. After the resource provider takes back their resources, the resource pool should have enough resources left to substitute these resources and offer the same level quality of service to the consumer. Otherwise the lack of resources would seriously impact on the quality of service and the credit of the Cloud Bank. So if we want to predict the liquidity risk, we should figure out what could cause lack of resources.

As shown in Figure 6.2, all the resource transactions of the Cloud Bank can be divided into four parts:

① Resource providers provide their resources to the Cloud Bank. The Cloud Bank gives an initial price of the resources and the saving interest rate, and then signs the saving contract with the Resource Providers.

② The Cloud Bank accepts the resource-renting applications from the Resource Consumers and gives the renting interest rate, then signs the renting contract with the Resource Consumers and offers the resources.

③ When the saving contract expires, the Cloud Bank should return the resources to the Resource Providers.

④ When the renting contract expires, the Cloud Bank will take back the resources that are offered to the Resource Consumers.

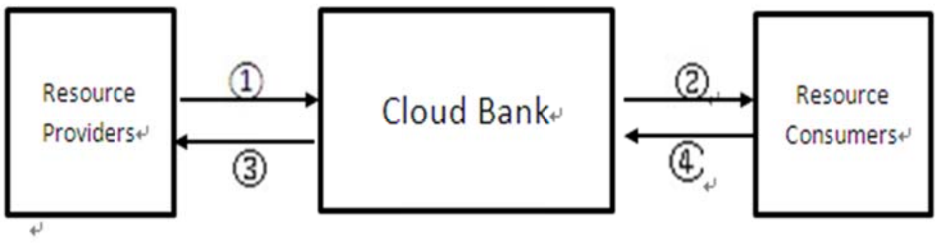


Figure 6. 2 The Resource Transactions in Cloud Bank

As mentioned above, renting resources to the consumer reduce the resources in the resource pool. It causes the lack of resources when provider takes their resources away. In order to predict the risk, we must monitor the Cloud Bank's loan business.

Here is an assumed situation as follows:

At the time t, the number of the remaining resources in the resource pool is L. There is a renting application. The resource number is AA and the renting time is T. During the renting time, the number of providers whose saving contracts would expire is m. For each Resource Provider i, the number of provided resources is S_i.

Based on this situation, the worst case is that all these m providers choose not to renew the contract. So at the time $(t + T)$, the number of the resources which would be taken away is$\sum_{i=1}^{m} S_i$. If

$$L - A > \sum_{i=1}^{m} S_i, \tag{12}$$

It means that the resource pool has enough resources to substitute during the term, so the liquidity risk would not happen. The Cloud Bank could agree to this application. If

$$L - A < \sum_{i=1}^{m} S_i \tag{13}$$

It means the Cloud Bank should not agree to this application. Although this approach can ensure that the liquidity risk would not appear during the term, but it is a static approach, limiting the size of the Cloud Bank's loan business.

In order to improve the approach, three dynamic situations should be added into the situation, which is mentioned above:

- During the term, there will be some new resources provided into the Cloud Bank, assume the number is A_d.

- During the term, the number of providers whose saving contracts would expire is m. For each Resource Provider i_ii, the number of the provided resources is S_i. The probability of choosing not to renew the saving contract is P_i.
- During the term, the number of the consumers whose renting contracts would expire is n. For each Resource Consumer j, the number of the using resources is S_j. The probability of choosing not to renew the renting contract is P_j.

According to these three dynamic situations, at time$(t + T)$, the number of resources which should be given back is $\sum_{i=1}^{m}(S_i \times P_i)$. The number of the resources in the resource pool is

$\mathrm{L} + A_d - \mathrm{A} + \sum_{j=1}^{n}(S_j \times P_j)$.

If the inequality

$$\mathrm{L} + A_d - \mathrm{A} + \sum_{j=1}^{n}(S_j \times P_j) > \sum_{i=1}^{m}(S_i \times P_i) \tag{14}$$

is satisfied, it means the resource pool has enough resources to substitute during the term, liquidity risk would not happen. Then the Cloud Bank could agree to this application. If (14) is not satisfied, it means the Cloud Bank could not agree to this application. (14) is called as the Predicting Inequality. The improved approach can ensure that the liquidity risk would not happen as long as (14) is satisfied.

But if the consumer of this application chooses to renew the renting contract when the contract expires, the Cloud Bank has no choice but to renew the contract. Because the environment of cloud computing is dynamic. The consumers always do not exactly know how much time is needed to finish their tasks. If their tasks are not finished when the contract expires, the consumer might choose to renew the contract. If the Cloud Bank chooses not to renew the contract and take the resources back, it might cause great loss of the consumer's benefit. The reputation of the Cloud Bank would be affected badly. As the Cloud Bank choosing to renew the contract, the renting time of this application is extended. If in the extended time, (14) is not satisfied, the liquidity risk may still happen. To predict the liquidity risk which happens because of the consumers renewing the renting contract, the Cloud Bank should monitor all the resources transactions and periodically check that whether the resource pool has enough resources T' is assumed as the time of one cycle. The number of the providers whose saving contracts would expire is assumed as m' in one cycle. For each Resource Provider$i$i, the number of the provided resources is S_i'. The probability of the Resource Provider choosing not to renew the saving contract is P_i'. The number of the consumers whose renting contracts would expire is n'. For each Resource Consumer j, the number of the using resources is S_j'. The probability of the Consumer choosing not to renew the renting contract is P_j'. The number of the new added resources is $A_d{}'$. The number of the remaining resources in the resource pool is L'. If the inequality

$$L' + A_d{}' + A' + \sum_{j=1}^{n'}(S_j' \times P_j') > \sum_{i=1}^{m'}(S_i' \times P_i') \qquad (15)$$

is satisfied, the liquidity risk may not happen in that cycle. If (15) is not satisfied, the liquidity risk may happen and the Cloud Bank should make the appropriate measures in time to manage the risk. (15) is called as the Monitor Inequality.

According to the prediction mentioned above, an improved Cloud Bank Model is set up, as shown in Figure 6.3:

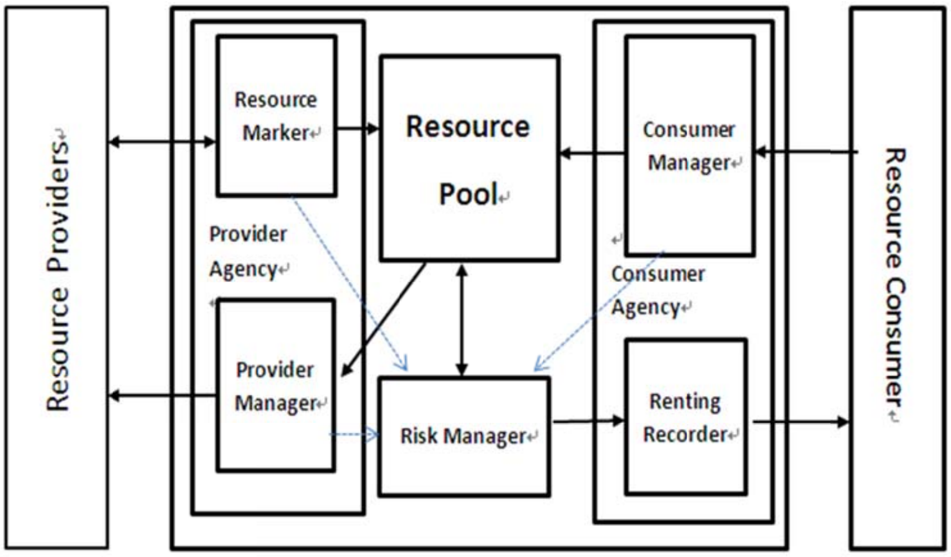


Figure 6. 3 The Liquidity Risk Prediction Model

In this model, the resource transactions can be divided into two parts: Provider-Bank transactions and Bank-Consumer transactions.

The process of the Provider-Bank transaction includes these operations as follows:

- Providers submit their providing applications to the Provider Agency (PA), which includes providers' personal information and their resources' information.
- Resource Marker (RM) will determine whether a provider has provided resources before in the Cloud Bank. If the provider does not have provided resources, RM will test its resources to see whether these resources are suitable for the environment of cloud computing. If these resources are suitable, RM will create a record including ID, number, contract time, credit record and so on. If these resources are not suitable, RM will refuse to add these resources into the resource pool. If the provider has provided resources before, RM will check its credit record and decide whether to agree to its application or not. After adding the resources into the resource pool, RM will update its history record in order to work out the parameter in (14) and the parameter in (15).
- When a deposit contract expires, the resource pool will return the resources to the providers. Provider Manager (PM) will record all the information about the resources that should be returned to the providers, including the provider's information. After updating the history record, PM should work out the parameter in (14) and in (15) for each provider. Then PM returns the resources to the providers.

The Bank-Consumer transaction includes these operations as follows:

- Consumers submit the renting applications to the Consumer Agency (CA), which includes the number of resources and the renting time.
- CA submits the information of each renting application to Risk Manager (RIM). RIM will decide whether to agree the applications.
- If RIM agrees to a renting application, the Renting Recorder (RR) will record all the information about the resources that will be rented to the consumers, including the consumers' information. The RR will rent resources to the consumers.
- When a renting contract expires, the consumers should submit the information about whether to renew the contract to CM. If the consumer chooses to renew the contract, CM will record the new renting applications and submit the information to RIM. After updating the history record, CM should work out the parameter in (14) and in (15) for each consumer j. Then CM adds the returned resources to the resource pool.
- RIM will periodically check whether the liquidity risk would appear in the future.

In the Predicting Inequality and the Monitor Inequality, there are some important parameters: A_d,P_i,P_j,$A_d{}'$, $P_i'$and P_j'. In the mathematical prediction models, Exponential Smoothing Model[129] is suitable to work out A_d and $A_d{}'$, Logistic Regression Model[130] is suitable to workout P_i, P_j, P_i' and P_j'.

The life cycle of Cloud Bank is divided into two phases. The speed of adding new resources into resource pool increases rapid in the initial phase and kept steady in the stable phase. When using the history data to predict the new added resources, the bigger the time span between the history time and the predicting time is, the more useless the history data is. Exponential Smoothing Model is a predicting model.

It can reduce the effect of the long time-span history data on the prediction result. The basic equation for Exponential Smoothing Model is

$$S_t = (a)Y_t + (1 - a)S_{t-1} \tag{16}$$

In (16), S_t is the smoothing value in cycle t, it means the predicting value in cycle t. Y_t is the actual value in cycle t, it means the actual value of the new added resources in cycle t; S_{t-1} is the smoothing value in cycle $t - 1$, $a$is the smoothing constant, its range is [0,1].

The computation of the parameter A_d of one providing application is shown as follows:

Step1: Initial work. Assume that this providing application's providing time is T; the whole history time of Cloud Bank is T_1. Evenly split the whole history time by T, the whole history time is divided into $\left({T_1}/{T}\right)$cycles; T is the serial number of the cycles. Let $\mathrm{t} = 1$, $S_{t-1} = 0$. Get the actual value Y_t from the history data of Cloud Bank, use t, $S_{t-1,}$ Y_t to work out S_t.

Step2: Let $S_{t-1} = S_t$, and make the value t as $(t + 1)$, get the actual value Y_t from the history data of Cloud Bank. Put S_{t-1}, Y_t into (16) to work out S_t.

Step3: Repeat Step 2 until $\mathrm{t} > \left({T_1}/{T}\right)$.

Step4: Let $A_d = \mathrm{S_t}$.

Then we get the value of A_d.

The computation of the parameter $A_d{}'$is similar to the value of A_d's.

When a contract expires, some providers or consumers choose to renew the contract, while others do not. The providers or consumers who make the same decision have some common features. The idea of Logistic Regression Model is to separate the providers or consumers who choose to renew their contract from those who make the different decision, and then study the two groups and find out the common features. According to the mapping between these common features and one provider's or consumer's own features, the probability can be worked out.

The basic equation for Logistic Regression Model is

$$\text{logit}\big(Pr(Y = 1|X)\big) = \beta_0 + \beta_1 X_1 + \cdots + \beta_m X_m \qquad (17)$$

In (17), $\text{logit}(x) = \ln(x/(1-x))$. $\text{Y} = 1$ means choosing not to renew the contract.X is the set of the features of a provider or a consumer.$Pr(Y = 1|X)$ is the probability of choosing not to renew the contract under the condition X. $\{X_i\}$ is the set of all the common features; $\{\beta_i\}$is the set of the model parameters; each $\beta_i$can reflect the importance of the X_i to the prediction.

The computation of the parameter Pi of a provider is shown as follows:

Step1: Study the history data and find out the set of the common features $\{X_i\}$.

Step2: Create the Logistic Regression equation based on the $\{X_i\}$:

$$\text{logit}\big(Pr(Y = 1|X)\big) = \beta_0 + \beta_1 X_1 + \cdots + \beta_i X_i \qquad (18)$$

Step3: Use the history data to work out each β_i.

Step4: For a certain provider, find out its set of features

$\{X_j\}\subseteq\{X_i\}$, let $\{X_p\} = \{X_i\} - \{X_j\}$.

For each $X \in \{X_j\}$, let $X = 1$.

For each $X \in \{X_p\}$, let $\text{X} = 0$.

Then put the value of $\{X_j\}$ and $\{X_p\}$ into (18).

After working out $(Y = 1|X)$, let $P_i = Pr(Y = 1|X)$.

Periodically update the history data, and then update the model parameters $\{\beta_i\}$ to keeps (18) effective.

The computations of the parameters are P_j , P_i' and P_j' are similar to P_i's.

6.2.3 The Strategy of the Risk Coping

The goal of the risk coping is to prevent the risks from arising. Before we consider how to prevent the risks, we first summarize what cause these risks. When the result of the risk prediction shows that some risks would arise in the next period of time, the Cloud Bank will start the strategy of risk coping.

As mentioned above, the risks in Cloud Bank are credit risk, liquidity risk, operational risk and other risk. In these risks, credit risk and liquidity risk are two most important risks that might often arise, so we are now focused on the two risks. Credit risk arises from the non-performance of a resource provider. It might cause the loss of the resources that would affect the services, which are provided to the consumers and at last cause the loss of the consumers' benefits. Liquidity risk arises from the Cloud Bank's inability to provide the services because resource providers withdraw too many resources. We can see that the two risks have one common feature: the Cloud Bank doesn't have enough resources to support its services to the consumers. So the best way to prevent these two risks from arising is getting enough resources in time.

In order to get enough resources in time, the strategy of the risk coping in Cloud Bank can be divided into two types: softness strategy and hardness strategy, as shown in Table 6.2:

Table6. 2 Cloud Bank Risks Tolerance Strategies Matrix

Risks Tolerance Strategies				
Softness Strategies	Hardness Strategies			
Pricing Strategies	Internal Tolerance	External Tolerance		
	Self-resources reserve	External-resources reserve	Resource leasing	Resource purchase

The softness strategy is suitable for those risks that would not arise rapidly. It uses the price leverage to adjust resource pricing and the interest rate. The strategy can adjust the composition of the resource pool. For example, raising the saving rate may attract more resource providers offering their resources to the Cloud Bank so that the Cloud Bank may get enough resources. The disadvantage is that the effect of pricing strategy is very slow. So pricing as the softness strategy is only suitable for those risks that will not arise recently, and the size of these risks is small, which means the number of the lack resources is small.

The hardness strategy is suitable for those risks that would arise soon. It includes self-resources reserve, external-resources reserve, resource leasing and resource purchase.

Self-resources reserve means the Cloud Bank should prepare a certain amount of resources, which are owned by the Cloud Bank. When no risk is about to arise, the state of these spare resources should be kept available. When some risk is about to arise, these spare resources will be the first choice as the supplements.

External-resources reserve will be used when the self-resources reserve is not enough to prevent some risk from arising. The resources of the external-resources reserve don't belong to the Cloud Bank, but there is a protocol between the owners and the Cloud Bank that can promise these resources being available whenever the Cloud Bank needs them.

If self-resources reserve and external-resources reserve are still not enough, the Cloud Bank will lease some resources or even buy some resources in order to support its service. The two strategies are called resource leasing and resource purchase.

As shown in Figure 6.4, Δ is used to represent the size of the risk. We assume that risk level is divided into ten levels. Bigger size of risk means higher risk level. Different risk levels need different risk coping strategy.

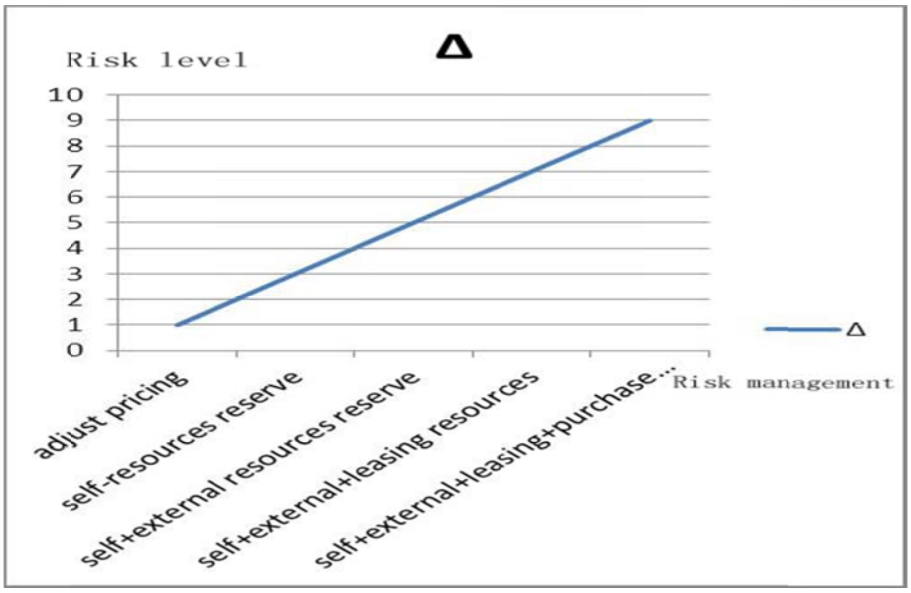


Figure 6. 4The Mapping between Risk coping Strategies and Risk Levels

6.3 Experiment Setup

According to the requirement of the risk prediction strategy, the whole experiment was divided into three main modules: Resource provider agency module, Risk prediction module and resource consumer agency modules. It shows in Figure 6.5

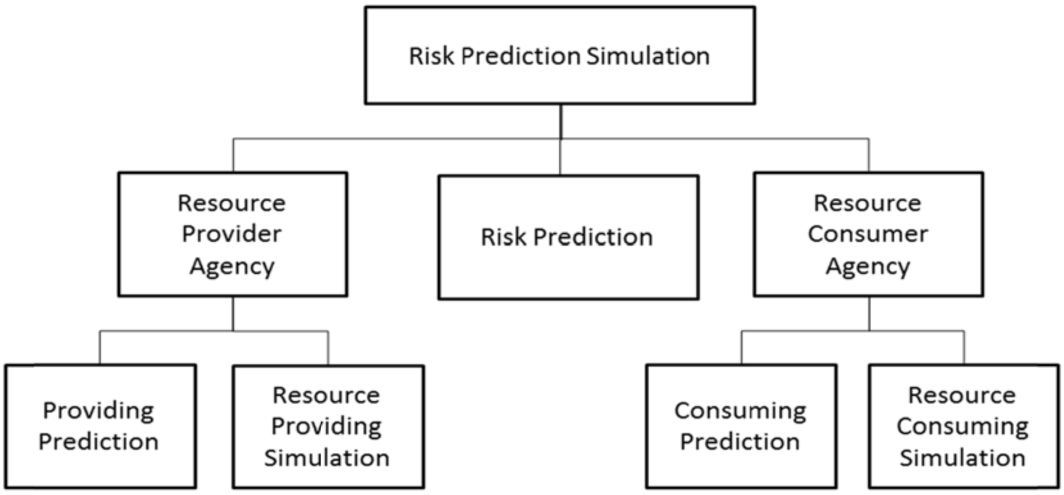


Figure 6. 5 Simulation Block Diagram

The resource provider agency module is mainly responsible to simulate the resource transaction process of the resource providers to provide resources to the cloud. The main functions are as follows:

- Record the type and quantity of resources provided by the resource providers, and joined the resource pool of the Cloud Bank.
- Forecast increases the types and quantities of resources within certain period of time. Calculations predict inequality in A_d values.
- To predict the resource providers renew the contract or not renew with the Cloud Bank.

The risk prediction module is mainly responsible simulate the Cloud Bank risk prediction module. The main functions are as follows:

- According to the status of the user's resource rent application, given the results of risk prediction.
- According to the state of the resource pool and resource modules as well as the use of resources module provides data, periodically to give a Cloud Bank resources trading environment risk prediction, then given the results of risk prediction.

The resource consumer agency module is mainly responsible for the simulation of the transaction process of the leasing resourcing between the resource consumer and the Cloud Bank. The main functions are as follows:

- According to the risk prediction results based on risk prediction module, response to the application resource. If the result present as risk, the module refuses to lease application, otherwise, the module record the type and quantity of the resource rent and synchronize to a resource pool
- To predict the resource users the option to renew

The resources provide risk prediction simulation is as follows:

- Initialization module status. Set the parameters of resources detection period t.
- Resource provider agency module provides some related information about the resource consumer and the resources, to determine the possibility of resource consumer renew the contract or not. Calculations predict inequality in P_i values.
- After the expiration of the contract, this module synchronizes resource pool data, and "returns" the resources to the providers.

The resources loan risk prediction simulation is as follows:

- Initialization module status. Set the parameters of resources detection period t.
- The resource consumer module simulates the instances of the resource loan, to offer related information about resource loan application.
- The risk prediction module gives the prediction result based on the above-related information.
- The resource consumer module given the appropriate measures based on the predicted results, and synchronizes the data of the resource pool. Calculations predict inequality in P_j values.
- After the expiration of the contract, recycling resources, data synchronization resource pool.

Some of the key codes as below:

```
publicclass Provider_Simulation {
public ArrayList<Resource_Info> ResourceAddHistory(ArrayList<Resource_Info>
resHistory){
        // Randomly generated the type of resource, the range is 1-4, as below:
        //1:CPU
        //2:Memory
        //3:HardDisk
        //4:Network
        long resType = Math.round(Math.random()*3+1);
        // Randomly generated the quantity of resource,the range is 100-1000
        long resNum = Math.round(Math.random()*900+100);
        // Randomly generated the trade,and match the resource taye and quantity
        Resource_Info resInfo = new Resource_Info();
        resInfo.setResource_Type((int)resType);
        resInfo.setResource_Num((int)resNum);
        resHistory.add(resInfo);
        return resHistory;
}

// Determination of the exponential smoothing model correction factor a
publicfloat Parameter_Determine(ArrayList<Resource_Info> ali){
        float a = 0.1f;
        float []b = null;
        float init,record = 0,sum=0,min=Float.MAX_VALUE;
        int i,swi=1,j = 0;
        init=(float)ali.get(0).getResource_Num();
        b[0]=init;
        while(swi==1&&j<ali.size()){
            sum=0;
            for(i=1;i<=ali.size();i++)
            {
// Obtained a correction parameter values corresponding to the variance of the number of s
                b[i]=a*((float)ali.get(i).getResource_Num())+(1-a)*b[i-1];
            }
            for(i=0;i<ali.size();i++)
            {

    sum+=(b[i]-(float)ali.get(i).getResource_Num())*(b[i]-(float)ali.get(i).getResource_Nu
m());
            }
// Correction parameters corresponding to the minimum variance prediction correction
parameters
            if(sum<=min){
                min=sum;
                record=a;
            }
            // Correction parameter a change gradient as 0.1
            a+=0.1f;
            j++;
        }
        a=record;
        return a;
    }

    // prediction process
    publicfloat[]ResourcePrediction(){
        int i=1;
// Correction parameters for each type of resource settings exponential smoothing model
        float Para_CPU,Para_Memory,Para_HardDisk,Para_Bandwidth,tem;
        // Each category of resources according to the final forecast of the exponential
smoothing model.
        float Ad_CPU,Ad_Memory,Ad_HardDisk,Ad_Bandwidth;
        float []Results = null;
        // Initialize various types of resources
        ArrayList<Resource_Info> aliHis = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliCPU = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliMemory = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliHardDisk = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliBandwidth = new ArrayList<Resource_Info>();
        while(i<=100){
            ResourceAddHistory(aliHis);
            i++;
        }
        // Classification of resources
        for(Resource_Info resInfo:aliHis){
            if(resInfo.getResource_Type()= =1){
                aliCPU.add(resInfo);
            }
            if(resInfo.getResource_Type()= =2){
                aliMemory.add(resInfo);
            }
            if(resInfo.getResource_Type()= =3){
                aliHardDisk.add(resInfo);
            }
            if(resInfo.getResource_Type()= =4){
                aliBandwidth.add(resInfo);
            }
        }
        // To determine the parameters of various resources correction
        Para_CPU = Parameter_Determine(aliCPU);
        Para_Memory = Parameter_Determine(aliMemory);
        Para_HardDisk = Parameter_Determine(aliHardDisk);
        Para_Bandwidth = Parameter_Determine(aliBandwidth);
        // Determine the predictions of the various resources
        // The predicted results of the CPU
        tem=(float)aliCPU.get(0).getResource_Num();
        for(i=1;i<aliCPU.size();i++){
            tem=(float)aliCPU.get(i).getResource_Num()*Para_CPU+(1-Para_CPU)*tem;
        }
        Ad_CPU=tem;
        // The predicted results of the memory
        tem=(float)aliMemory.get(0).getResource_Num();
        for(i=1;i<aliMemory.size();i++){

    tem=(float)aliMemory.get(i).getResource_Num()*Para_Memory+(1-Para_Memory)*te
m;
        }
        Ad_Memory=tem;
        // The predicted results of the HardDisk
        tem=(float)aliHardDisk.get(0).getResource_Num();
        for(i=1;i<aliHardDisk.size();i++){
    tem=(float)aliHardDisk.get(i).getResource_Num()*Para_HardDisk+(1-Para_HardDisk)*
tem;
        }
```

```
            Ad_HardDisk=tem;
            // The predicted results of the Bandwidth
            tem=(float)aliBandwidth.get(0).getResource_Num();
            for(i=1;i<aliBandwidth.size();i++){

      tem=(float)aliBandwidth.get(i).getResource_Num()*Para_Bandwidth+(1-Para_Bandwid
th)*tem;
            }
            Ad_Bandwidth=tem;
            Results[0]=Ad_CPU;
            Results[1]=Ad_Memory;
            Results[2]=Ad_HardDisk;
            Results[3]=Ad_Bandwidth;
            return Results;
      }
}
```

6.4 Analysis of Experimental Results

This simulation process, the setting is fixed the detection cycle is 30 seconds. The experimental simulation time is 10 minutes. To simulate ten times transactions between the resource providers and the Cloud Bank transactions. Summary statistics result shown in Table 6.3.

Table 6. 3 the Result of Summary Statistics

Time Node	Resource pool state			Resource trading event					Risk prediction	Final Result
	CPU (Unit)	Memory (G)	Hard Disk (G)	Trade Type	CPU (Piece)	Memory (G)	Hard Disk (G)	Contract Time(s) (S)		
0:00	50	100	2000							
0:30	50	100	2000	1	4	10	200	50	No	1
1:00	54	110	2200	2	20	28	1130	60	No	1
1:30	50	100	2000	2	24	53	576	30	No	1
2:00	50	100	2000	2	18	76	588	50	No	1
2:30	50	100	2000	1	36	98	1545	40	No	1
3:00	86	198	3545	2	8	87	803	100	No	1
3:30	42	13	1197	1	38	4	624	70	No	1
4:00	80	17	1821	1	14	23	661	90	No	1
4:30	102	127	3285	2	48	58	1371	120	No	1
5:00	16	65	1290	2	2	43	1251	140	Yes	2
5:30	2	57	629	1	24	47	366	80	No	1
6:00	26	104	1059	2	48	6	1730	90	Yes	2
6:30	26	104	1059	1	39	69	1004	60	No	1
7:00	89	169	3004	1	17	49	773	20	No	1
7:30	106	218	3777	1	11	49	369	20	No	1
8:00	100	218	3373	2	49	67	1840	100	No	1
8:30	51	151	1533	1	42	71	1018	30	No	1
9:00	43	104	1178	1	29	87	1797	20	No	1
9:30	1	33	160	2	8	67	1715	50	Yes	2
10:00	1	33	160	2	36	54	1236	70	Yes	2

Notes: The resource provider, present as “1”, the resource consumer, present as “2”

As shows in table 6.3, the trade been divided as two types: The resource provider, present as “1”, the resource consumer, present as “2”. The risk predication has two results. The “No” presents no risk and “Yes” presents have risk. Finally, the Cloud Bank has final risk predication result as: “1” presents available to trade and “2” presents no respond to trade.

As shows in the table 6.3, at the time 5:00,6:00,9:30and 10:00will result in resource trading risks. The experiments show that the proposed risk prediction strategy is effective.

6.5 Summary

The environment of cloud computing is distributed and dynamic. There are lots of uncertainties, which can cause risks in the resource transactions. So risk prevention is a very important part in the whole Cloud Bank model. Cloud Bank's risks can be divided into the following categories: credit risk, liquidity risk, operational risk, and other risks and so on.

This chapter first analysis the risk mitigation and coping in real commercial bank. After that, this thesis studies the risk mitigation in the Cloud Bank. Based on that analysis, this paper gives the classification of the risk in the Cloud Bank. There are Credit Risk, Liquidity Risk Operational Risk and Other Risk.

For the features of Cloud Bank reason, the credit risk never cause by the consumer, because the consumer do not really take any resource physically. The Credit risk arises from non-performance by a resource provider. The Liquidity risk arises from the Cloud Bank's inability to provide the services because resource providers withdraw too many resources. If the resource pool does not have enough resources to replace these removed resources, then the quality of service of the Cloud Bank would inevitably decrease. The benefits of the consumers would be aggrieved.

The Cloud Bank's operational risk happens when the Cloud Bank's internal operational management and control mechanisms fail. Other risks mean those risks might have bad effect on the Cloud Bank. For the reason of topic of operational risk and other risk are too big to management in this thesis and they will conduct research in the future. The paper just focuses on the credit risk and liquidity risk.

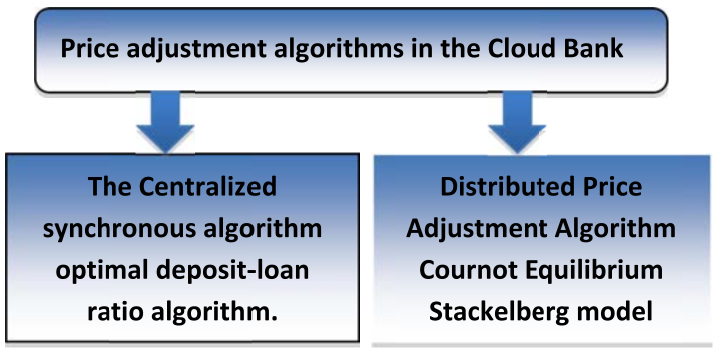
After analysing the risks arising in Cloud Bank, this thesis found that prediction is the most important part in risk prevention. The contribution of this part was divided the risk prediction into two parts: risk prediction and risk coping. The predicting strategies of all risks are presented. The predicting of liquidity risk is divided into two parts: predicting before renting and periodical prediction. Two inequalities are presented to judge whether the risks would arise and the size of the risks. In the two inequalities, two mathematical prediction models are used to determine the parameters. At last the whole liquidity risk predicting model is presented. This model can help Cloud Bank predicting the liquidity risk and prevent it from arising.

This part is divided the risk coping into two types: softness strategy and hardness strategy. The softness strategy is suitable for those risks that would not arise recently. It uses the price leverage to adjust resource pricing and the interest rate. The hardness strategy is suitable for those risks that would arise soon. It includes self-resources reserve, external-resources reserve, resource leasing and resource purchase.

Finally, using CloudSim and experiment to test the risk mitigation and coping related algorithms.

Chapter 7 Research Approaches for the Pricing Scheme of the Cloud Bank in the Price lifecycle

This chapter presents the need to initialize the price for the treasury of the bank- the resource pool during the construction of our Cloud Bank. Within this model, the price adjustment algorithm of resources is divided into two categories. There are centralized synchronous algorithm and distributed price adjustment algorithm. Initially, the centralized synchronous algorithm uses the notion of total supply and demand balance to simultaneously adjust prices for all resources. The distributed price adjustment algorithm then first classifies the resource, then, based on the situation of each type of resources' supply and demand, adjusts the price to achieve balance between resource supply and demand.


Price adjustment algorithms in the Cloud Bank
The Centralized synchronous algorithm optimal deposit-loan ratio algorithm.
Distributed Price Adjustment Algorithm Cournot Equilibrium Stackelberg model

7.1 The Centralized synchronous algorithm

In this Cloud Bank model, the centralized synchronous algorithm this thesis used was an optimal deposit-loan ratio algorithm.

7.1.1 the theory of optimal deposit-loan algorithm

The deposit-loan ratio is based on the ratio of the bank's total deposits to its loans. The resource provider's equivalent to the savers of commercial banks, corresponding resource consumers are similar to borrower, meanwhile, the Cloud Bank layer plays the commercial bank's deposit-loan transaction role. The Cloud Bank pays out to the depositors certain interest, according to the storage time, and charges the loan fee according to certain interest and time used to execute tasks. In order to stimulate resource providers depositing resources into the Cloud Bank and thereby allowing borrow to take a loan of resources from the bank, realize win-win situation of supply and demand sides. Like wise to achieve supply and demand balance and ensure a maxim profit for our hypothetical Cloud Bank operators, we introduced an optimal deposit-loan ratio theory used in commercial banks to guide the Cloud Banks' daily management, which allows the adjustment of the deposit-loan ratio in order to achieve an optimal rate of both deposits and loans. Using these parameters, more lenders are not necessarily better for the bank. The reality of the situation in a cloud environment is that it can be quite hard to guarantee that all the depositors and lenders will follow the policies outlined in the service contract.

A similar difference between cloud resources and physical assets actually managed by commercial banks is that cloud resources—being electronic—will occasionally fail during variety situations. As a consequence, the Cloud Bank must maintain backups in order to perform their contractual obligations. In this environment, a higher rate of conversion from deposits to not necessarily good. Instead, there is an optimal ratio of deposits to loans that can theoretically be achieved and there by maximize the benefits, but only under this optimal ratio. Taking a cue from real banks, we attempted to find an "optimal ratio of deposits and loans" that might potentially achieve our ends and ensure optimal services.

7.1.2 The resource management model based on optimal deposit-loan algorithm

Assume that b is the sensitive coefficient of loan L to loan interest rate i_1, and d is the sensitive coefficient of deposit D to deposit interest rates i_2 where b 、 d are constants, $(b < 0, d > 0)$.[112]

Namely,

$\partial L/\partial i_1 = b$,

$\partial D/\partial i_2 = d$

Obviously, Cloud Banks oriented towards making profit derive income from the interest on virtual loans and must make expenditures on deposit interest. The model then would look as follows:

$$R = i_1 L - i_2 D - C_1(L) - C_2(i_1, L) - C_3(D) - C_4(i_2, D) \tag{1}$$

where R is the Cloud Banks' profit, C_1 (supervisory cost of loans) varies based on the size of the loans. The larger the loan's scale, the larger the supervisory cost. C_2 as loans default cost, is affected by the loans interest rate and loans scale, and whether is isolated moves with loans interest rate and the size of loans. C_3 (Cloud Banks' operating cost) mainly depends on the Cloud Banks' resources scale. Total deposits decide the bank resources size, and total deposit and operation cost whether is isolated move. C_4 (Depositors default cost) is mainly influenced by deposit interest rates and total deposit, the lower deposit interest rate and the bigger total savings, the greater risk of default. C is total cost, namely $C = C_1 + C_2 + C_3 + C_4$. Assuming r is the deposit-loan ratio, the model changes into Equation (1), R of D ,i_1 and d i_2 can respectively be derived as:

$$\frac{\partial R}{\partial D} = i_1 r - i_2 - r\left(\frac{\partial C_R}{\partial rD}\right) - r\left(\frac{\partial C_2}{\partial rD}\right) - \frac{\partial C_3}{\partial D} - \frac{\partial C_4}{\partial D}$$

(2)

$$\frac{\partial R}{\partial i_1} = rD + i_1\, r\left(\frac{\partial R}{\partial i_1}\right) - i_2\, r\left(\frac{\partial R}{\partial i_1}\right) - r\,\left(\frac{\partial C_1}{\partial rD}\right)\left(\frac{\partial D}{\partial i_1}\right) - \frac{\partial C_2}{\partial i_1} - r\left(\frac{\partial C_2}{\partial rD}\right)\left(\frac{\partial D}{\partial i_1}\right)$$
$$-\left(\frac{\partial C_3}{\partial rD}\right)\left(\frac{\partial D}{\partial i_1}\right) - \left(\frac{\partial C_4}{\partial D}\right)\left(\frac{\partial D}{\partial i_1}\right) \quad (3)$$

$$\frac{\partial R}{\partial i_2} = i_1\, r\left(\frac{\partial D}{\partial i_2}\right) - D - i_2\left(\frac{\partial D}{\partial i_2}\right) - r\,\left(\frac{\partial C_1}{\partial rD}\right)\left(\frac{\partial D}{\partial i_2}\right) - \frac{\partial C_2}{\partial i_1} - r\left(\frac{\partial D}{\partial i_2}\right)$$
$$-\left(\frac{\partial C_3}{\partial rD}\right)\left(\frac{\partial D}{\partial i_1}\right) - \left(\frac{\partial C_4}{\partial i_2}\right) - \left(\frac{\partial C_4}{\partial D}\right)\left(\frac{\partial D}{\partial i_2}\right) \quad (4)$$

Make (2), (3), and (4) is equal to 0 can get

$$r = (i_2 + \partial C/\partial D)/i_1 \quad (5)$$

$$(\tfrac{b}{r})(ri_1 - i_2 - \tfrac{\partial C}{\partial D}) + rD - (\tfrac{\partial C_2}{\partial i_1}) = 0 \quad (6)$$

$$d(ri_1 - i_2 - \tfrac{\partial C}{\partial D}) - D - (\tfrac{\partial C_4}{\partial i_2}) = 0 \quad (7)$$

We can find the deciding factor of r is deposit interest rates, loans rates and deposit scale marginal cost from (5). The r, deposit interest rates and deposit scale marginal cost is whether isolated change, reverse with loans rates. Deposit scale is also the important factors that affect the marginal cost. We can imagine if the bank sonly deposit is 1 dollar, adding another 1 dollar deposit will increase marginal costs to a much greater degree than if bank's deposits initial deposit is 10,000 dollars and another 1 dollar deposit is added. This concept is keenly illustrated by the deformation of (7).

$$\frac{\partial C}{\partial D} = -\frac{D}{d} + ri_1 - i_2 - (\tfrac{1}{d})\ (\tfrac{\partial C_4}{\partial i_2}) \quad (8)$$

From (8) it is clear that the marginal costs to the bank for deposits area deposit scale monadic linear function, which decreases when deposits increase. The $\partial C_4/\partial i_2$ is a sensitive coefficient, which is caused by saver's default risk and the cost to deposit interest rates. Since resources providers enter into an agreement with the grid bank, their own resources often sit idle and in some cases retrieving their resources is a small possibility. Put more succinctly, the possibility of $\partial C_4/\partial i_2$ approximate to 0. D is i_1 i1 and i_2 function, therefore r decision factors is the deposit interest rates and loans rates.

Via (6) and (7) and assumption (3), we can give the equation (9). As the loan default cost is $\partial C_2/\partial i_1$:

$$r = \left\{\frac{\partial C_2}{\partial i_1} + [(\partial C_2/\partial i_1)^2 + 4(b/d)D^2]^2\right\}/2D \tag{9}$$

The larger the r means that the deposit to loan conversion rate is higher, but with investment of diminishing marginal returns rule, in a certain spot, r value of Cloud Bank operational stability of marginal utility to offset its negative to the capital formation of contribution margin, so as to achieve the optimal value. To (9) for deformation get:

$$\partial C_2/\partial i_1 = rD + (b/d)(D/r) \tag{10}$$

Take $L = rD$ into (10)

$(\partial C_2/\partial i_1)/L = 1 + (b/d)(1/r^2)$ into

Because$(\partial C_2/\partial i_1)/L > 0$, so $r^2 \geq -b/d$. From the above analysis we can determine $\partial C_2/\partial i_1$ and the Cloud Bank's operational stability is negatively correlated, the block bank earnings growth L and capital formation (i.e., resource deposit) is positively correlated, then this can promote bank earnings growth if $(\partial C_2/\partial i_1)/L$ achieves a minimum value. Succinctly, Cloud Bank management stands to make the greatest gains in such a growth state. Given that r and $(\partial C_2/\partial i_1)/L$ is positively correlated, the smaller the r, the smaller$(\partial C_2/\partial i_1)/L\left(\partial C_2/\partial i_1\right)/L$, so when $(\partial C_2/\partial i_1)/L$ takes a minimum value, r should take$(-b/d)^{1/2}$, highlighting that the optimal deposit-loan ratio is the negative square root of the sensitive coefficient, which can be understood as the loan-to-loan interest rate divided by the sensitive coefficient of deposit to deposit interest rates.

7.1.3 Single resource pricing underlying the Cloud Bank model

Given the flexibility of the Cloud Bank model, it is possible to introduce a variety of different situations that use resource management and economic trade models, such as the commodity market model, bargaining model, bidding model, monopoly model, and etc[132].

After comparing the two different strategies—the commodity market strategy and the auction strategy—Dusak found the commodity market model more suitable for controlling the trade and distribution of resources, and during this evaluation developed a equation to calculate the price of CPU and disk storage.[133]Unlike traditional computing, in the cloud users do not use a single CPU or memory, and accordingly a calculation equation require a combination of two factors to derive the basic resource price. First, it is crucial to consider several factors on the in measuring CPU value: the number of slots provided by resource (slots), the relative costs (RC), and the unit time period (unit time). In this case the RC stands for the CPU, disk and other single resource device's costs relative to the same types of equipment on the market. In the event of a new resource provider—like our Cloud Bank model—at the first it is necessary to evaluate their resources and arrive at a measurement score (S) before calculating the relative cost while recording the price of the same types to the market price (MP).

$$\mathrm{RC} = \frac{\mathrm{MPS}}{\mathrm{L}} \tag{11}$$

L means a price equilibrium parameter. CRPs are willing to provide some CPU interfaces that can stable use to Cloud Bank, the number of interfaces depends on CRPs' CPU processing speed and a CPU ratio that they willing to provide. The equation is as follows:

$$\mathrm{cpu\_price} = \frac{\mathrm{RC}/_{\mathrm{unit\_time}}}{\mathrm{slots}} \tag{12}$$

cpu_price denotes the relative cost that the CPU provides value service to the Cloud Bank for a unit time expressed by each interface. For example, CPU relative cost RC for 10 dollars with a unit time of 1hour and two available slots, then the cpu_pricewould be that the average cost of each interface—in this case, about 5 dollars an hour.

The method of measuring disk storage value is similar to that used in measuring CPU value, but the parameter changes, as capacity indicates storage space provided by CRP and $storage_price$is the relative cost at which the disk(s) provide value for use in Cloud Bank for a unit time expressed by each G. The equation is as follows:

$$\text{storage\_price} = \frac{\text{RC}/\text{unit\_time}}{\text{capacity}} \qquad (13)$$

For example, the relative costs for the disk are 100 dollars with a unit time of 1hourproviding a total of up to 10G storage space. In this example, the $storage_{price}$ would be calculated by the average cost of each G, or about10 dollars an hour/G.

As noted earlier, in the cloud users are not limited by limited hardware resources, so combining the CPU prices and disk prices is needed to obtain the price equation of a single resource unit cost per unit time:

$$\text{cost\_price} = \text{m} \times \text{cpu\_price} + \text{n} \times \text{storage\_price} \qquad (14)$$

Among these, the parameters m and n measures different types of price parameters in proportion to resources.

7.1.4 of the optimal deposit-loan algorithm

As discussed above, computing resources and storage resources have an initial price. This section details our use of a distributed strategy for resource price adjustment that includes the price of the CPU and DISK. In our scenario, the ideal of algorithm is:

1. Set acceptable minimum deposit interest rate when the cloud resource provider deposit their resource into the bank, when the deposit interest rate is lower than its minimum value, the provider will leave the Cloud Bank, thus reducing the number of resource providers.
2. Correspondingly, set an acceptable highest lending rate that allows cloud resource consumers to borrow resource from the bank, when the lending rate is higher than its maximum value, the lender will likewise leave the Cloud Bank, thus reducing then number of resource consumers.
3. By setting a time interval, at the beginning of every time interval, it is possible to adjust the deposit/lending rate to control the total deposits and loans and to archive the optimal deposit-loan interest rate. This will create a process that can maximize the interests of all involved parties and move the deposit-loan ratio towards optimal conditions.

7.1.4.1 Algorithm for price adjustment of the computing resources in the resource pool.

1. Initializei_1, i_2; ($i_1$lending rate, $i_2$deposit interest rate).

2. Initialize the price for every single CPU computing power as $\mathrm{CPU}(i)$.
3. Initialize the total price for all CPU computing power in the resource pool as $\mathrm{D} = \sum CPU\ (i)$;
4. Initialize the total price for all CPU computing power borrowed from the resource pool as $\mathrm{L} = \sum CPU\ (j)$;
5. Initialize the random variable a, b,
6. **While** abs$\left(\frac{L}{R} - r\right) < m$ // r is optimal deposit-loan interest rate and m is a small floating-point.
7. **If**$(r - L/D) > m$ **do**
8. $i_1 = i_1 - a || i_2 = i_2 - b$ // lower the deposit-loan interest rate
9. Setting all the acceptable maximum loan interest rates in descending order and taking all the resource back leading to a lending rate price lower than i_1,then calculate $\mathrm{L/D}$;
10. Setting all the acceptable minimum deposit interest rates in ascending order and remove all the resource from the bank for which the deposit interest rate is higher than i_2,then calculate$\mathrm{L/D}$;
11. **If**$\left(\frac{L}{D} - r\right) > m$ **do;**
12. $i_1 = i_1 + a || i_2 = i_2 + b$ //higher the deposit or loan interest rate.
13. Consumer agent according to the value ofi_1 notices the consumer the price $i_2$which is acceptable and asks the consumer renting the resource or not. If the consumer wants to rent the resource, then transfer to the Cloud Bank, otherwise, the consumer leaves the bank. Calculate $\mathrm{L/D}$ again.
14. Provider agent according to the value of i_2 notices the consumer the price i_1 which is either acceptable or not and asks the provider to deposit the resource or not. If the provider wants to deposit the resource, proceed to examination and registration, then transfer to the Cloud Bank, otherwise, the provider leaves the bank. Calculate $\mathrm{L/D}$ again.
15. Repeat steps (6) - (14) to move to the next cycle of price adjustment.

7.1.4.2 Algorithm for the price adjustment of the storage resources in the resource pool

1. Initialize i_1, i_2; ($i_1$lending rate, $i_2$deposit interest rate).
2. Initialize the price for every single DISK storage capacity as $\mathrm{DISK}(i)$.
3. Initialize the total price for all DISK storage capacity in the resource pool as $\mathrm{D} = \sum DISK\ (i)$.
4. Initialize the total price for all DISK storage capacity borrowed from the resource pool as $\mathrm{L} = \sum DISK\ (j)$.
5. Initialize the random variable$a,$ □^{a}
6. **While** abs$\left(\frac{L}{R} - r\right) < m$ //r is optimal deposit-loan interest rate and m is a quite small floating-point.
7. **If**$\left(\frac{L}{D} - r\right) > m$**do**
8. $i_1 = i_1 - a || i_2 = i_2 - b$// , lower the deposit-loan interest rate;
9. Setting all the acceptable maximum loan interest rates in descending order, and taking all resources back for which lending rate price is lower than i_1,then calculate $\mathrm{L/D}$;
10. Set all the acceptable minimum deposit interest rates in ascending order and remove all the resource from the bank for which the deposit interest rate is higher than i_2, then calculate $\mathrm{L/D}$;
11. **If**$\left(\frac{L}{D} - r\right) > m$ **do;**
12. $i_1 = i_1 + a || i_2 = i_2 + b$ // higher the deposit or loan interest rate;
13. Consumer agents according to the value of$i_1$notes the consumer the price i_2 which is acceptable or not and prompts the consumer to rent the resource or not. If the consumer wants to rent the resource, then transfer to the Cloud Bank, otherwise, the consumer leaves the bank. Calculate $\mathrm{L/D}$ again.
14. Provider agent according to the value of$i_2$notes the consumer the price i_1 which is acceptable or not and then prompt the provider to deposit the resource or not. If the provider wants to deposit the resource, proceed to examination and registration, and then transfer to the Cloud Bank, otherwise, the provider leaves the bank. Calculate

L/D again;

15. Repeat steps (6) - (14) and proceed to the next cycle of price adjustment.

On the whole, when the deposit interest rates decline, the total amount of deposits$D$ in a period of time will be reduced, otherwise, it will increase. When the loan interest rates decline, the total amount of loans L in a period of time will be reduced, otherwise, it will increase.

7.2 Distributed Price Adjustment Algorithm

At the beginning of the transaction, all the Cloud Resource Provider (CRP) has nothing to do with any given consumers, but the bank must give a price for the CRP. The Cournot model will lead to a state of price during this initial time. After initializing the price of the resource, the Cloud Bank will begin to run smoothly. Given the nature of these cloud resources, all the CRPs are in a dynamic state, as CRPs will occasionally join or quit the Cloud Bank resource pool. The Cloud Bank then needs to have a way to price all new entering resources according to the present situation. In this instance, the Stackelberg model will lead the price to an optimized state over the running time.

7.2.1 Pricing Scheme of Cloud Resources in the Initial Stage

7.2.1.1The Cournot Equilibrium in Cloud Computing

In the beginning, the initial price of the underlying resources is decided by the interaction of CRPs, which has nothing to do with consumers. The resources providers only consider the characteristic of a monopoly model, and under this, the price of resources is decided by the interaction among resource providers. This situation ultimately leads all the different CRPs to the same price strategies.

Two commonly monopoly models worth noting are the Cournot model[134]and the Bertrandmodel[135]. In the Cournot Model, CRPs must guess the price under the production, but in the Bertrand Model the price is used as a strategy by CRPs to infer monopoly pricing, for which the solution is Nash equilibrium in regard to price. Taking into account the available resources afforded to cloud computing in accordance with a given unit of time—which is in connection with the amount of available resources—the Cloud Bank is able to then introduce a Cournot Equilibrium Model to begin equalizing the price.

Definition 1 - Cournot Equilibrium: In the environment of a market economy, the participants all know one another's output, and each participant determines their own output in order to maximize profits, leading to the output of any given participants remaining roughly unchanged [136].The solution to such equilibrium is a Nash equilibrium, in which any given participant maximizes their own interests while the others' interests also maximize. In doing so, users within a cloud computing environment can maximize their interests under the Cournot equilibrium theory.

Definition 2 -Cloud Computing Cournot Equilibrium: In the cloud computing environment, resources provider know the resource volume provided by other resources providers, and as such choose their what volume of their own resources can be made available in order to make the amount of resources help maximize their own individual interests.

7.2.1.2 Mapping the physical resources into the cloud resource pool

Resource providers and consumers within the cloud environment have their own requirements and strategies to succeed as well as different operational definitions of success. In this present study, a paramount concern is defining the provider role, which we have done in order investigate the pricing mechanisms at work in the proposed Cloud Banking model.[137]

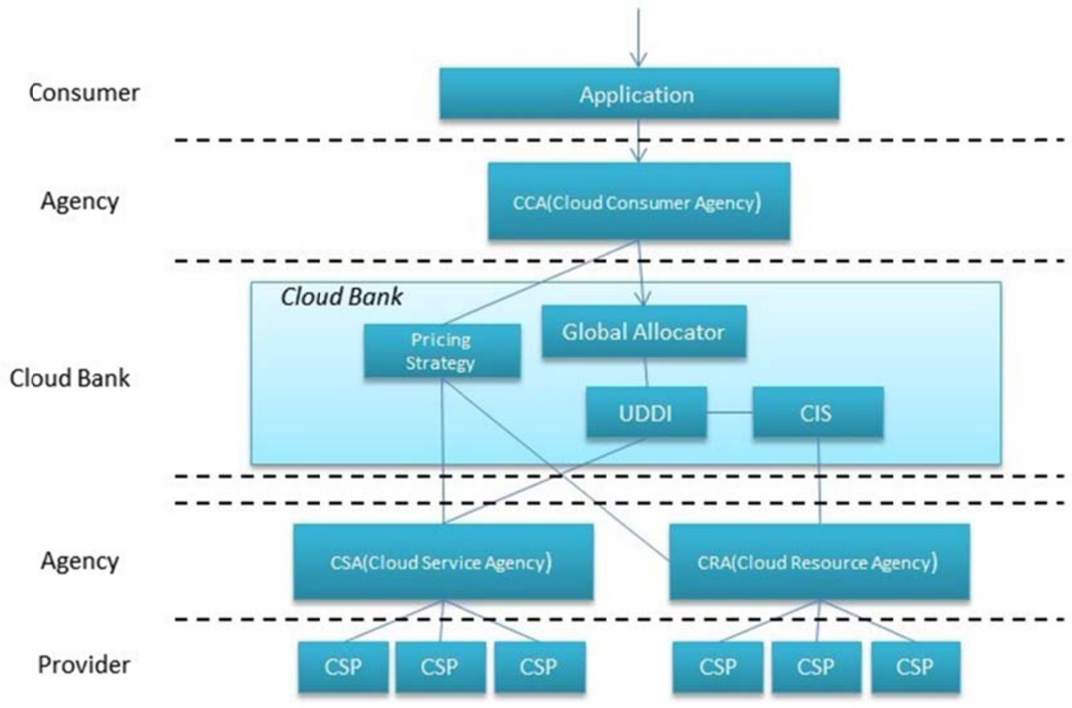


Figure 7. 2 The Cloud Bank Resource Architecture Based on Economics

In figure 7.1, Cloud Bank consumer proposes their application requests and describes the service requests such as completion time, budget. The Cloud Bank consumer agency responsible for receiving the consumer's request and then submit this request to the Cloud Bank, it will instead of the consumer establish SLA with the service provider. The Cloud Bank meanwhile is responsible for providing resource management services and pricing strategy. Resource management services is in charge of dynamic monitor resources and services, furthermore, it release the resources and services information to the cloud information server (CIS) and universal description discovery and integration (UDDI); Trading services (Pricing strategy) is responsible for consultation with the cloud service provider(CSP) and cloud resource provider(CRP) to define a reasonable price model, record the service condition of resources and services, fee to user, pay for the CSP and CRP.

Cloud service agency (CSA) is in charge of collecting the relevant service information provided by service providers and then registers this information to the UDDI; it is also responsible for calls related services and establishes SLA with Cloud Bank consumer agency(CCA) after CCA search service, which meet consumer’s needs. The Cloud resource agency (CRA) is responsible for collection relevant resource information (resource preliminary pricing, resource description, etc.) provided by resource providers, and then registering this information to CIS.

7.2.1.3 Using the Cournot Equilibrium to initialize Cloud resources prices

Under the Cloud Bank environment, resources providers are defined as producers, who will provide a competitive service in order to attract resource consumers and maximally utilize currently idle resources; users who have application requests are defined as consumers, who will generally want the resources to cost as little as possible and be as convenient to access as possible. In our improved model, the original provider is broken down further into two roles: Cloud Bank resource providers (CRP) and cloud service providers (CSP). CRPs provide physical resources, e.g., CPU, memory, storage, and so on (Fig7.3). CSPs mean while provide specific types of services such as distributed computing service, high-throughput computing services, data-intensive computing services, etc.

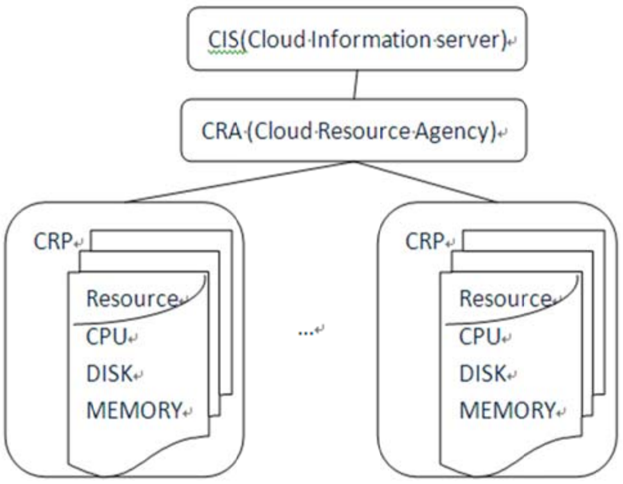


Figure 7. 3 Logical Structure of Cloud Bank Resource Providers

Resources provided by CRPs need to be understood in abstract descriptions, allowing the concept of price to measure specific resources and calculate the basic price under for each resource under the pricing strategy. Because of the competition between CRPs, in order to guarantee price equilibrium on the premise of maximizing the interests of all CRPs, we introduced the Cournot model as a potential solution.

Taking into account the work done to date at this juncture in testing and building the Cloud Bank model, in particular calculating the individual resource price provided by each resource providers, we opted to introduce the concept of equilibrium based price before arriving at a final price. Since the traditional Cournot equilibrium only takes into account two players, in a Cloud Bank environment, this model would need to still operate in roughly the same manner while it was expanded to reach the goals of more participants. To create these conditions, There are many resource providers and new providers will join in at any time.

Resource providers compete with each other in order to obtain more market share, assuming that they have already known the volume of resources provided by other providers, they determine their own volume of resources to maximize their own interests.

The products provided by different resource providers are indistinguishable, while different consumers choose different resources to obtain the same effect.

$\mathrm{A_i}$ means the amount of resource provided by the i_{th} resource provider,$\mathrm{cost_price_i}$represent the price of single resource provided by the i_{th} resource provider, the total price of resource expressed as C. In the case of multiple participants, multiple resource prices set to

$\mathrm{C} = \{\mathrm{C_1}, \mathrm{C_{2,}} \dots \mathrm{C_n}\}$, $\mathrm{C_i}$ denoting the cost prices of the i_{th} resources. Then:

$$\mathrm{C_i} = \mathrm{A_i cost\_price_i} \tag{15}$$

In the Cloud Bank environment, there is an interaction between resource providers, so calculating the initial price P based on the basic price, which is a function of $P(A)$of the total amount of resources where AA is the total amount of resources and $\mathrm{A} = \sum_1^n \mathrm{A_i}$.

Each resource i_{th} providers takes into consideration its amount of resources that can impact the total output prices, which can be described as:

$$\frac{\partial \mathrm{Q}}{\partial \mathrm{A_i}} = \frac{\mathrm{dP}}{\mathrm{dA}} \times \frac{\partial \mathrm{A}}{\partial \mathrm{A_i}} \tag{16}$$

The amount of resources provided by the i_{th} provider are $\mathrm{A_i}$, a single resource price is $P(A)$, the cost price is $\mathrm{C_i}$, therefore its revenue function as follows:

$$\mathrm{\pi_i} = \mathrm{P(A)} \times \mathrm{A_i}\text{-}\mathrm{C_i} \quad \mathrm{i} \geq 2 \tag{17}$$

In order to maximize the benefits of the i_{th} resource providers, using $\mathrm{A_i}$ from the (17), gives the following equation:

$$\frac{\mathrm{d\pi_i}}{\mathrm{dA_i}} = \mathrm{A_i}\frac{\mathrm{dP}}{\mathrm{dA}} + \mathrm{P(A)}\text{-}\frac{\mathrm{dC_i}}{\mathrm{dA_i}} = 0 \tag{18}$$

or simplified:

$$\mathrm{A_i} = \frac{\acute{\mathrm{C}}_1\text{-}\mathrm{P}}{\acute{\mathrm{P}}} \tag{19}$$

$\mathrm{A_i}$ denotes the amount of resource provided by the i_{th} resource provider in equation (19), and during this time their interests become maximized. As a result of the price impacted by supply and demand, $P(A)$ in connection with $\mathrm{A_i}$ which is the amount of resources available provided by the i_{th} resource provider. If the specific type function of $P(A)$is already known, we can find the amount of resources provided by the i_{th} resource provider when the profits of the i_{th}ith provider are maximizing.

Taking into account the competition and interaction between the various participants, we must first consider the specific relationship between the i_{th} and the j_{th} provider. Deriving $\mathrm{A_i}$ from the type (8) obtains a reaction function slope between i and j, wherein temporarilythe number of resource provider is n, where$j \neq i$ and $j \in (1,2,\ldots,n)$. After simplification,

$$\frac{\mathrm{dA_i}}{\mathrm{dA_j}} = \frac{(\mathrm{P}\text{-}\acute{\mathrm{C}}_\mathrm{i})\ddot{\mathrm{P}}\text{-}\acute{\mathrm{P}}^2}{2\acute{\mathrm{P}}^2\text{-}\ddot{\mathrm{C}}_1\dot{\mathrm{P}}\text{-}(\mathrm{P}\text{-}\acute{\mathrm{C}}_\mathrm{i})\ddot{\mathrm{P}}} \tag{20}$$

$\mathrm{C_i}$ is a linear function of $\mathrm{A_i}$, so there are $\ddot{\mathrm{C}} = 0$. But the price function $P(A)$ cannot determine its form, so there may be a balance of M ($M \geq 1$) or non-existent equilibrium in the game process. If $P(A)$ is a linear function of $\mathrm{A_i}$, the following assumption can be made: $\mathrm{P(A)} = \sum_1^n \alpha_\mathrm{i}\mathrm{A_i} + \mathrm{K}$, which contains the server parameters, such α_i as balanced parameters and K which indicates the t relationship between the price and the amount of resources. Consequently, if $\ddot{\mathrm{P}} = 0$, then the equation (20) can be simplified as the following:

$$\frac{\mathrm{dA_i}}{\mathrm{dA_j}} = \text{-}\frac{\acute{\mathrm{P}}^2}{2\acute{\mathrm{P}}^2} = \text{-}\frac{1}{2} \tag{21}$$

At this point, the response function can be illustrated in Fig7.4.

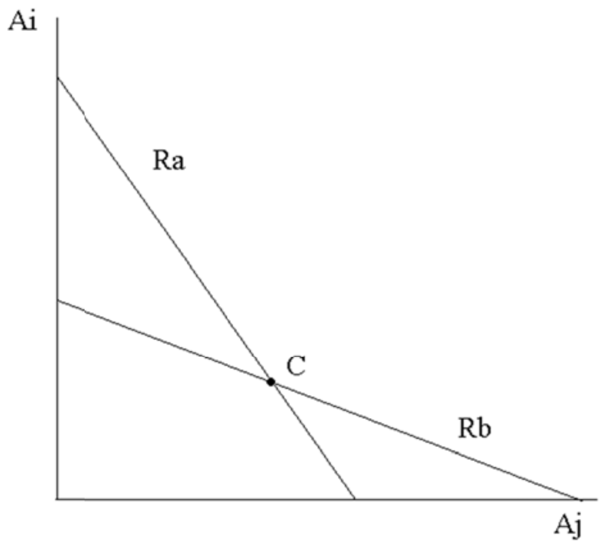


Figure 7. 4 The Response Curve of Cournot Equilibrium

The first step is to assume that the equilibrium is symmetric, leaving

$A_i = A_j(i \neq j \text{ and } j \in \{1,2,3,\ldots,n\})$, at this time each participant faces the same conditions, which can be substituted as:

$$A_i = \frac{\acute{c}-P}{\acute{P}} = \frac{\text{cost\_price}_i - \sum_1^n \alpha_j A_j + K}{\sum_1^n \alpha_j} \tag{22}$$

This equation can be further simplified as:

$$A_i = \frac{\text{cost\_price}_i + K}{2\sum_1^n \alpha_j} \tag{23}$$

So P (A)'s result as follow:

$$P(A) = \sum_1^n (o_i \frac{\text{cost\_price}_i + K}{2\sum_1^n \alpha_j}) + K \tag{24}$$

7.2.2 Pricing scheme of the cloud resources in a stable stage

7.2.2.1 The concept of the Stackelberg Leadership Model in cloud computing transactions

When the transaction states reaches the stable stage, the resources' prices will change as random producers' either join or quit, so we opted to use the Stackelberg Leadership Model to guide the pricing strategy.

The Stackelberg model is a strategic game in economics in which the leader firm moves first and then follower firms move sequentially. In terms of game theory, the players of this game include a leader—sometimes referred to as the Market Leader—and a follower that both compete on quantity. To sustain the equilibrium of the game, there are some constraints. The leader must know *ex ante* that the follower observes his action. The follower must have no means of committing to a future non-Stackelberg follower action, and the leader must know this. Indeed, if the 'follower' could commit to a Stackelberg leader action and the 'leader' knew this; the leader's best response would be to play a Stackelberg follower action[138].

7.2.2.2 Using the Stackelberg Leadership Model to adjust Cloud resources prices

Using Stackelberg leadership model, the Cloud Bank (i.e., the oligopoly) as the leader (E_1), the random producers as the follower (E_i)

In this understanding, the hypotheses are as follows:

There are many random resource producers, denoted $E_i(i = 2, \ldots, n, n \geq 2)$. These producersare able toprovide several resources while new producers and resources enter or drop out as time goes.

Producer E_1 provides resources first, though they do not know what output quantity strategy that $E_i(i = 2, \ldots, n)$ will choose; the $E_i(i = 2, \ldots, n)$ act later and their resource output according to E_1's production. They do not however know the output quantity strategy between $E_i(i = 2, \ldots, n)$. The profits then is a function of resources provided by all producers; each producer must make an output strategic decision in order to maximize their profits.

One important note worth mentioning here is that the resources from different providers are in distinguishable, meaning that no matter what resources the user selects the ultimate effect is the same.

A_i represents the amount of resource provided by the i_{th} resource producer, while p_i represent the price of single resource provided by the ith resource producer which is cost_price mentioned in section 7.1.3.1, the total price of resource expressed as C. In the case of multiple participants, multiple resource prices are set to $C = \{C_1, C_{2,} \ldots, C_n\}$, with C_i representing the cost prices of the i_{th} resources, leading to:

$$C_i = A_i\, P_i(\mathrm{A}) \qquad (25)$$

On the basis of basic price, we need to compute the initial price of resources P, which is the inverse demand function P(A) of total resources where A represents the number of all resources, A= $\sum_1^n A_i$. Presuming that

P(A)=P($\sum_1^n A_i$)=$a - b\sum_1^n A_i$ $(b > 0, 0 < c < a)$ (26)

in the basic balance of supply and demand. A represents the number of the social demand per unit time when the resources are free, while b reflects the sensitivity of market needs impact to price, and the profit of E_i is π_i.

In this hypothetical case, E_1 provides resources first, E_i follow E_1, and the profit of E_1 is:

$\pi_1 = A_1 P(\mathrm{A}) - C_1$

(27)

While the profit of E_i is

$\pi_i = A_i P(\mathrm{A}) - C_i (i \geq 2)$, (28)

When considering E_1 as the leader, they must choose the output of resources$A_1 \geq 0$,other producers observe A_1 then choose their output $A_1 (i = 2, \dots, n)$ according to A_1. Subsequently, the strategy of E_1 is to select A_1 while the strategy of E_i is a function of

$S_f: Q_1 \to Q_f$ (29)

Where $Q_1$is the output of E_1, and Q_f is the sum of E_i's output.

This is a dynamic game model whereby all parties act in a perfect order and use perfect information. Using a reverse solution we are able to compute the sub game perfect Nash equilibrium.

To begin, consider the second stage of this game, with a given outputA_1. How does E_i select their optimum output $A_i (i = 2, \dots, n)$(i=2,, n) to get maximize profits:

$Max\pi_i = A_i P(\sum_{j=1}^n A_j) - C_i, (i = 2, \dots, n.)$ (30)

, Afterwards, substitute the inverse demand function and cost function into equation (30) we are able to compute the first order optimal condition, omitting the process of matrix algorithms, and arrive at the optimal decision of E_i,

$\dot{A_\iota} = \frac{a - P_i - A_1}{2b(n-1)} (i = 2, \dots, n.)$ (31)

Now we consider the first stage of this game, because E_1 predicted E_i would select the optimal decision $\dot{A}_i$ based on the equation (31) to maximize profits, E_1's problem can be described as:

$$Max\pi_1 = A_i P\left(\sum_2^n \dot{A}_i + A_1\right) - C_1 \tag{32}$$

(9)Next, substituting equation(31) into equation(32), consider the inverse demand function and cost function at the same time, we can compute the first order optimal condition of equation(31),and omitting the calculation process, arrive at the optimal decision of E_1:

$$\dot{A}_1 = \frac{a-p_1}{2b} \tag{33}$$

Substituting equation (33) into equation (32) yields the optimal decision of E_i:

$$\dot{A}_i = \frac{a-p_i}{4b(n-1)}\ ,\ (i = 2, \dots, n) \tag{34}$$

In effect:

$$P(A) = P(\sum_1^n A_i) = a - b\sum_1^n A_i = a - b(\dot{A}_1 + \sum_2^n \dot{A}_i) \tag{35}$$

Next, substituting $\dot{A}_1$ and $\dot{A}_i$ into equation (33) yields:

$$P(A) = \frac{3a}{4} + \frac{p_1}{2} - \frac{\sum_2^n p_i}{4(n-1)} \tag{36}$$

Finally, we arrive at the initial price of resources $P(A)$. According to the equation, we then find that the initial price is influenced by demand quantity and supply quantity of resources, and moreover that it varies with them.

7.3 The Service Level Agreement (SLA) of the Cloud Bank Model

The Cloud Bank is essentially an agent platform of resource management. To control the use and receipt of computing resources from, and by the Cloud Bank requires the use of a service level agreement (SLA). As computing resources are provided in the form of services, Quality of Service (QoS) mechanisms are introduced to solve problems in the quality of service. This study presents a framework for SLAs in the Cloud Bank model (CBSLA) with QoS attributes added to the CBSLA so that it can become a key aspect to facilitate exchanges among the participants. In order to describe services, the CBSLA can be used to reflect the price.

7.3.1 Why we need CBSLA

QoS is one of the substantial aspects that distinguish the differences between various service providers. In order to assure the consumer that they can get the service they pay for while also obligating the provider to achieve its service promises, the Cloud Bank can act as a third party that helps both the provider and consumer come to an agreement based on the requirements of the consumer's demand and the provider's supply.

At the core of this interaction, an SLA is a negotiated agreement between two parties wherein one is the consumer and the other is the resource provider which guarantees the basics components of the services offered. The service guarantees spell out precisely what transactions need to be executed and how well they should be executed. The SLA itself records a common understanding about services, priorities, responsibilities, guarantees, and warranties. Additionally, the SLA may specify the levels of availability, serviceability, performance, operation, or other attributes of the service, such as billing[139]

Despite the many uses of the SLA, perhaps among the most important components of the SLA are the QoS attributes. SLA reports must provide a guarantee that the QoS is being monitored and that the Cloud Bank can deal with any accident that may happen. Failure to do so would inextricably undermine the relationship between the different involved parties as well as the Cloud Bank itself, which serves as an intermediary third-party facilitator.

Because the demands of consumer are complex, a simple "measure and trigger" process may not work for enforcing the SLA. Aside from this potential difference, which arises due to the complex nature of the services provided in the Cloud Bank, most SLAs are natural language documents and accordingly must be provisioned and monitored manually. Obviously, doing this on a scale needed in a cloud environment like ours would be quite troublesome. In order to automate provision and monitor the QoS parameters, we used a particular SLA template that includes automatically processed fields in an otherwise natural language-written SLA. To carry out the SLA, we used the principle of market economy regarding price.

7.3.2 The CBSLA Framework

The CBSLA is designed to formally capture the service level agreements in the Cloud Bank environment. The life cycle process of CBSLA is shown in Fig7.4, which identifies five main processes[140].

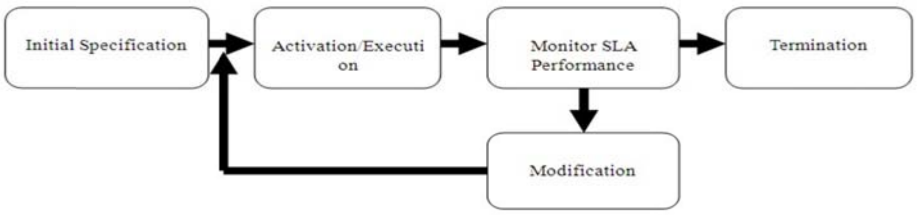


Figure 7. 5 Lifecycle of the SLA

The service level agreement (SLA) life cycle to govern a service level agreement from being initially identified, through to being activated, and, eventually, terminated when it is no longer required. Initial specification of the SLA is important to identify every aspect of service, such as identifying business requirements, defining service level goal and SLA metrics. Monitoring the performance of the SLA performance allows the Cloud Bank to review current terms and metrics and update them based on evolving requirements. Once the SLA needs to improve, the process will send it back to the initial stage.

A CBSLA is a service definition that defines an agreed set of service parameters and a manner in which to the aforementioned service. There are two parts of CBSLA, one is Resource Provider-SLA (RP-SLA), which is signed by Cloud Bank and resource provider, and the Resource Consumer-SLA (RC-SLA) which is signed by Cloud Bank and the resource consumer.

Fig7.5 illustrates general architecture of Cloud Bank service level management (CBSLA), wherein the provider places their resource into resource pool after quantification and grading by the Service Level Management (SLM), after which the RP-SLA contract will be signed. The resources in resource pool are provided as different levels of services according to the resource grade determined by the SLM. Likewise, the consumer signs the RC-SLA contract with Cloud Bank when they need services.

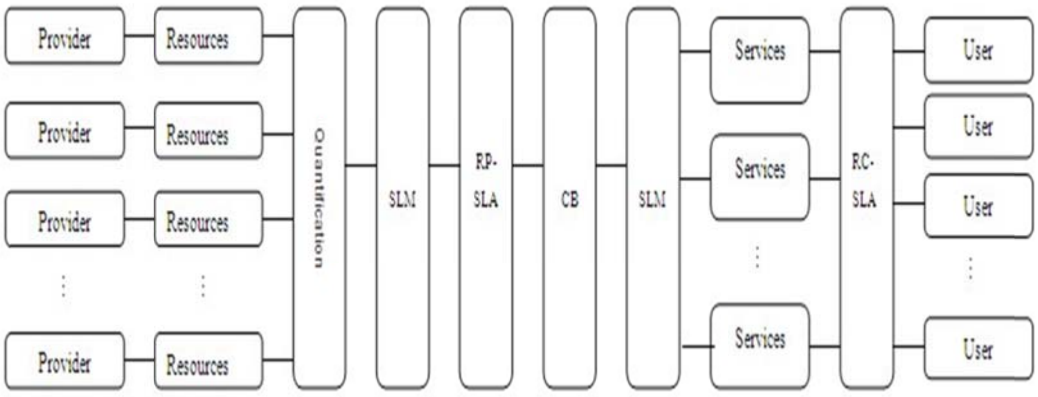


Figure 7. 6 General Architecture of the Cloud Bank Service Level Management

In this model, each of the activities represents an operation on resources and services. Consequently, an RP-SLA can be attached to each of the activities to represent the agreement between Cloud Bank and provider. Similarly, an RC-SLA can be attached to the overall process to represent the agreement between Cloud Bank and the end-user.

7.3.3 Signature process of the CBSLA contract

The whole signature process of CBSLA contract is outlinedinFig7.6:

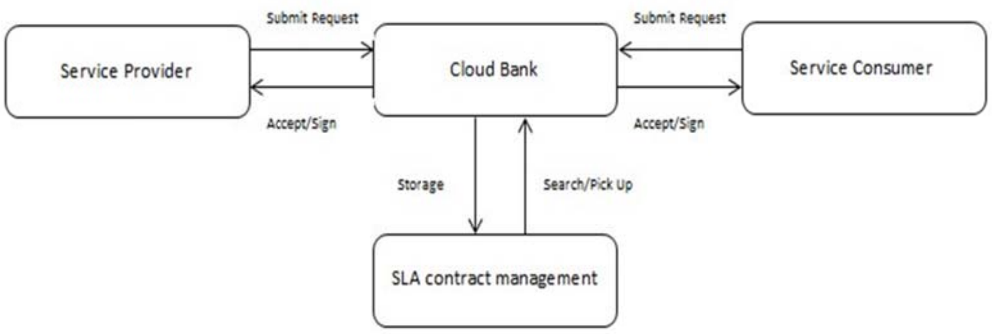


Figure 7. 7 Whole Signature Process of the CBSLA Contract

7.3.3.1 Signature process of the RP-SLA contract

In the Cloud Bank model, the Bank will define most of the content of a CBSLA, and the provider and consumer must then agree or decline as they see fit, or provide additional information. In all cases, pre-specified, fixed information and negotiable elements as well as their choices can be captured in a CBSLA template. A template can be published in a registry such as UDDI. Fig7.7 illustrates the signature process of the RP-SLA contract.

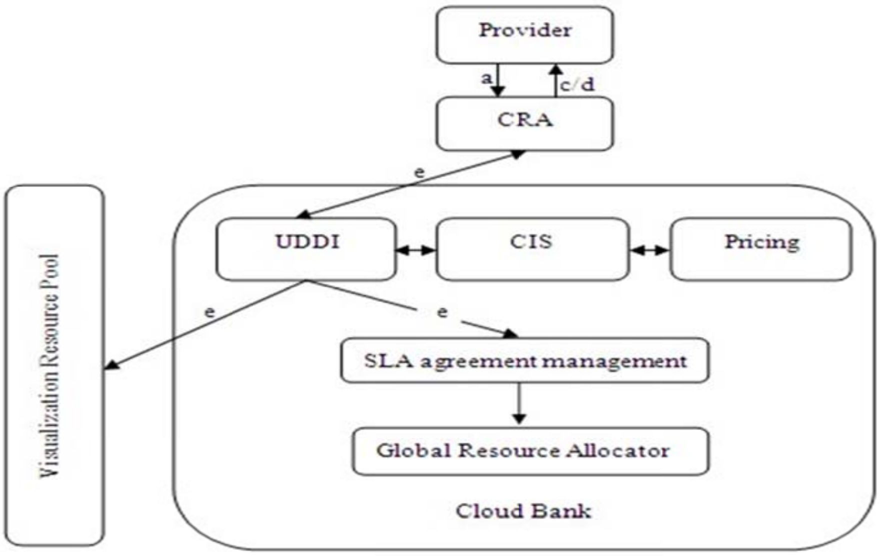


Figure 7. 8 Signature Process of the RP-SLA Contract

a) Resource provider send an RP-SLA contract request named the Request-RP-SLA to the Cloud Bank where the contract request is received and analysed by CRA.
b) The CRA obtains the actual parameters of the resources from provider by quantification and then compares it to the parameters extracted from the Request-CBSLA. If the actual parameters match with those of the Request-CBSLA or the disparities between them are acceptable, the CRA will inform Cloud Bank that this contract can be signed, after executing Step d. If the disparity is unacceptable, then execute Step c.
c) The CRA send a new RP-SLA contract with the actual parameters to the resource provider to negotiate and then move back and execute Step a.
d) The Cloud Bank sends the final RP-SLA contract to the resource provider to sign.
e) Once the RP-SLA contract is signed, the RP-SLA contract will be managed by SLA contract management and the information of these resources will recorded by UDDI and Global Resource Allocator.

7.3.3.2 Signature process of the RC-SLA contract

While the Cloud Bank defines the parameters of any given service, including definition of CBSLA measurement methods, management may offer a choice to the consumer as to the details of the guarantees. Fig6.8 illustrates the signature process of the RC-SLA contract.

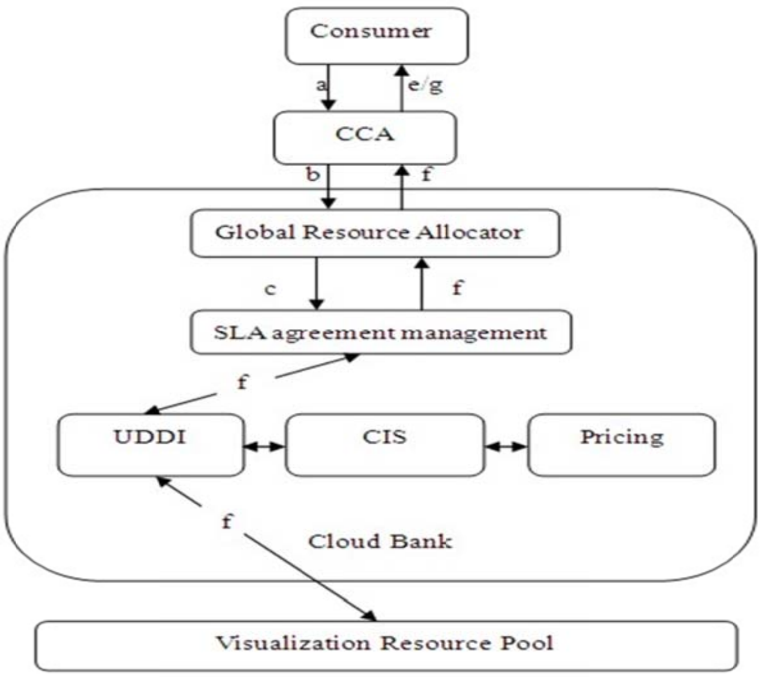


Figure 7. 9 Signature Process of the RC-SLA Contract

a) Similar to the RP-SLA, the resource consumer sends an RC-SLA contract request named the Request-RC-SLA to the Cloud Bank, after which the contract request is received and analysed by the CCA.
b) The CCA submits the parameters to Global Resource Allocator once analyzed.
c) Based on these parameters, Global Resource Allocator divides the whole task into several subtasks and then computes the demand of each subtask and submits them to SLA contract management.
d) The SLA contract management searches its own database to find the most suitable RP-SLA. If successful, execute Step f, and if not send the feedback to CCA and execute Step e.
e) The CCA sends a new RC-SLA contract with the modified parameters to the consumer to negotiate, after which execute Step a.
f) UDDI finds the service and submits it to the CCA based on the RP-SLA.
g) The Cloud Bank replaces resource providers to establish an RC-SLA with consumers according to the RP-SLA of each subtask.

h) Finally, SLA contract management will manage the RC-SLA contract too. After that, the SLA monitoring will run to ensure that the total QoS of each subtask is equal to the QoS of the whole task.

7.3.4 Generation of the CBSLA

After quantifying the cloud resources, it is possible to add them to the SLA and let it become certified among parties that have the resources transaction. The SLA mentioned in this thesis is a combination of the above two projects, which was put in the application of management and monitoring QoS property in the three-lever Cloud Bank structure. This in turn can be used to describe the application mission with QoS property, and through SLA, to bind the application and resources and enter into a contract bound to be strengthened. Resource consumers and the consumer agent will set up an SLA as long as services sets have applied for the physical resources to meet the QoS characteristics. All of the SLA and its property values can then be quantified and measured via similar methods as mentioned in the extant literature[141]. The local resource management and network resource management devices can measure the corresponding property and value of the QoS in the Cloud Bank model, and in doing so determine whether these resources and the QoS property values are in agreement with the SLA. Related resources at the SLA parameters between different layers of different manifestations through a number of inter-layer mappings been converted.

7.4 Summary

By extending the Cloud Bank model we developed, in this section we presented how to develop a pricing strategy that corresponds to different features of the Cloud Bank throughout different stages of time. Because there are different kinds of resource providers, such as oligopolies providing large-scale resources and retail entities that provide random resources, a flexible but coherent strategy is absolutely necessary in making this Cloud Bank model viable. To maximize profits for each party participating in the Bank, this thesis opted to base pricing strategy based on the Cournot equilibrium, allowing bank to adjust the price dynamically during the initial stage, and later moving into a Stackelberg Leadership Model based pricing strategy based during the stable stage.

Chapter 8 Research Approaches for the Pareto Optimality Based Scheduling

Theoretically, after an appropriate pricing mechanism has been established and put into use, gradually more consumers will find it advantageous to enter into the Cloud Bank. Once this growth in popularity occurs, resource scheduling becomes the key issue in maintaining stability and operation of the Cloud Bank. In this chapter, we accordingly describe a resource scheduling mechanism based on economic theory.

The cloud computing multi-resources market model is based on full market competition. The supply and the demand for resources ultimately decide the resources' price. When the quantity of both supplies and demands are equal, an equilibrium price is reached. According to the economic general equilibrium theory, the resource distribution that satisfies the equilibrium price is most reasonable and most effective.

The multi-commodity market model provides the most superior deployment of resources. As for our purposes it is the most ideal and the simplest economic model. Since the cloud computing environment is complex and the potential scope is huge, among the resource providers there is direct competition to mutually restrict one another. As such, the resource price cannot be completely decided by the supplies, given that there is essentially a non-cooperation game of a fixed price strategy existing among cloud computing resource providers. To move past this obstacle, the Cloud Bank model uses the Pareto Optimality Theory to give a concrete, definite resource distribution strategy and related algorithm.

8.1 The concept of Pareto optimality

Pareto efficiency, or Pareto optimality, is a concept in economics with strong applications in both engineering and social sciences. The term is named after Vilfredo Pareto, an Italian economist who used the concept in his studies of economic efficiency and income distribution.

Given an initial allocation of goods among a set of individuals, a change to the allocation that makes at least one individual better off without making any other individual worse off is called a Pareto improvement. When no further Pareto improvements can be made in this system, an allocation is defined as being "Pareto efficient" or "Pareto Optimal."

8.1.1 Pareto Optimality

A Pareto optimal state is generally considered to be one with an optimal allocation of social resources, or an effective standard of resources allocation. In this state if a person or people's benefit were altered, the situation of some another people is better, if we want them to be better, we need the situation of some people be worse. In welfare economics, Pareto optimal state is mainly discussed in the consumption and production. This matches cloud resource management in the relationship of the resource providers and resource users. In Cloud Bank, the result of resource scheduling among tasks is to get the Pareto optimality to make the system running smoothly.

8.1.2 Pareto Improvement

A Pareto improvement is defined as where in some economic circumstances an improvement to some people's the welfare or satisfaction can be made by without reducing any other person's satisfaction or benefits. Two definitions are worth considering.

Definition 1 -Consumption utility U: Consumers gain some benefits through consumption, and the value of that assumption is based on consumption quantity of each commodity.

Definition 2 -Consumer preference is a fixed value of utility U.

A Pareto improvement is an approach or a procedure that can make resource distribution among resource consumers enter into a Pareto Optimal state. In our Cloud Bank, this can correspond to the resource scheduling mechanism, and accordingly we have to design an algorithm that can achieve this mechanism.

8.1.2.1 Analysis of Pareto Improvement

Detailed analysis is as follows:

1) Two consumers: A, B;
2) Two kinds of goods: X, Y;
3) Two consumers' preferences are definite, and the consumer's utility functions are:

$$U_A = U_A(X_A, Y_A)$$

$$U_B = U_B(X_B, Y_B)$$

$$X_A + X_B = X_0$$

$$Y_A + Y_B = Y_0$$

Where U_A is the consumer A's utility, X_A, Y_A are the consumption numbers of two kinds by A, it's the same to B, X_0 and Y_0 are the total consumption numbers for two goods. This situation can be described in Figure 8.1

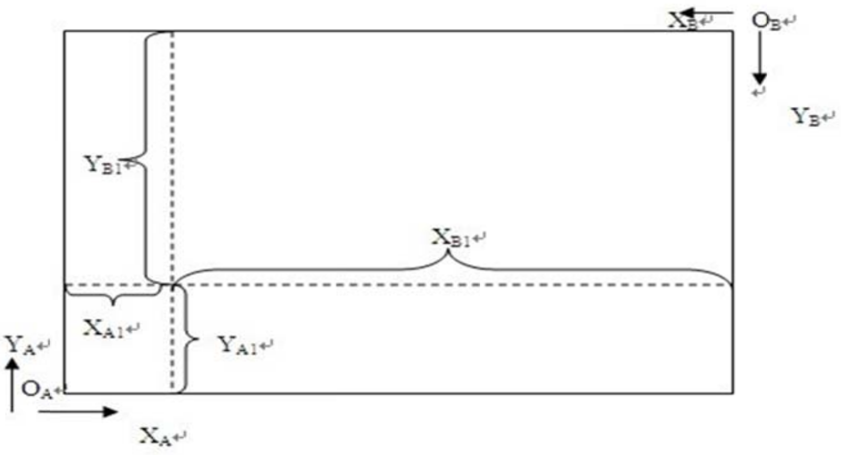


Figure 8. 1 Distribution of Consumer Goods[142]

The horizontal direction is X consumption (horizontal length $X_A + X_B = X_0$). The vertical direction is Y consumption (vertical length $Y_A + Y_B = Y_0$). Any point indicates a kind of allocation between the two consumers regarding these goods, for example, point $a(X_{A1}Y_{A1}, X_{B1}Y_{B1})$.

The indifference curve is a chain of two commodities consumed by consumers who have the same preferences, meaning that different combinations will get the same utility, each point in the figure represents a different allocation. This curve is shown in Figure 8.2.

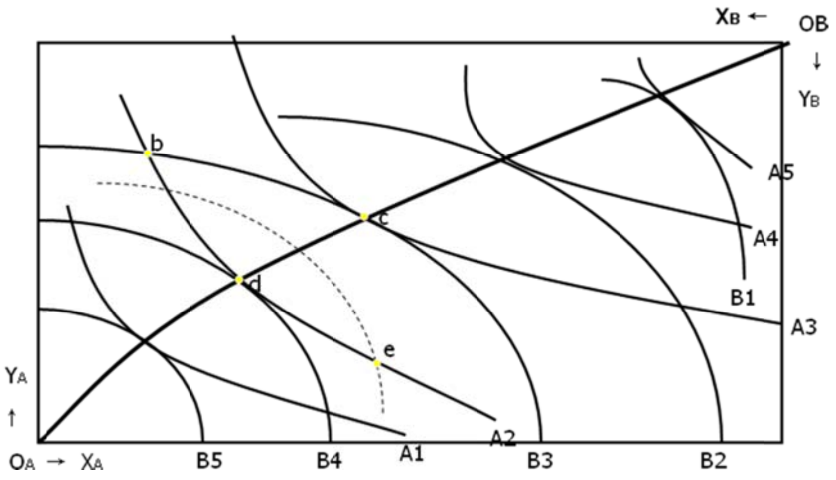


Figure 8. 2 Consumer Indifference Curve[144]

There are numerous indifference curves, and different curves represent different levels of utility, the further away from origin, the higher the utility is.

For consumer A:

$U_{A1} < U_{A2} < U_{A3} < U_{A4} < U_{A5}$,

For consumer B:

$U_{B1} < U_{B2} < U_{B3} < U_{B4} < U_{B5}$.

Definition 3: Marginal rate of substitution (MRS): the slope of the indifference curve tangent to the absolute value. That is to say, we premise it on a maintainable utility level, if a consumer buys one more X goods, he must give up a certain amount of Y goods, and it means that two kinds of goods consumption have a relationship of substitution. Above all, under the case of Pareto optimality state, any two arbitrary kinds of goods' MRS are equal.

As is shown in Figure 8.2, point b is the connection of indifference curve A2 and B3, but this point is not Pareto optimal state, when consumer B remain utility level in B3, consumer A's utility level can be increased to A3 ($U_{A3} > U_{A2}$) to point c, at this point A's utility is maximum, meaning that it is Pareto optimal state. If A's utility level remains in A2, the utility level of consumer B could be increased to B4 ($U_{B4} > U_{B3}$) to point d, which is a Pareto optimal state.

Likewise in going from c to d, it reaches Pareto optimal state. The line link O_A to O_B is a collection of Pareto optimal states, we call this collection an efficiency curve, and different points in the curve represent different resource allocations, e.g., from point c to point d, consumer B increases utility by decreasing A's utility.

8.2 Cloud Banks achieve optimal resources allocation by Pareto theory

Cloud Bank package resources to services in order to provide to the outside, these services are corresponding to the above commodities, services users are corresponding to the above consumers. Assumptions:

1) Two resources users: A, B;
2) Two services: X, Y;
3) Two persons' preferences are fixed, consumers' utility functions are

   $U_A = U_A(X_A, Y_A)$

$$U_B = U_B(X_B, Y_B)$$

$$X_A + X_B = X_0$$

$$Y_A + Y_B = Y_0 \ .$$

Known: X_0, Y_0, U_{A0}, work out MAX (U_B)

Construction Lagrange functions:

$$U_{B=} U_B\,(X_B, Y_B\,) + \lambda_0(U_A(X_A, Y_A) - U_{A0}) + \lambda_1(X_A + X_B - X_0) + \lambda_2(Y_A + Y_B - Y_0)$$

Derivation functions:

$$\frac{\partial U_B{}^*}{\partial X_B} = \frac{\partial U_B}{\partial X_B} + \lambda_1 = 0 \tag{1}$$

$$\frac{\partial U_B{}^*}{\partial X_B} = \frac{\partial U_B}{\partial Y_B} + \lambda_2 = 0 \tag{2}$$

$$\frac{\partial U_B{}^*}{\partial X_A} = \lambda_0 \frac{\partial U_B}{\partial X_A} + \lambda_1 = 0 \tag{3}$$

$$\frac{\partial U_B{}^*}{\partial X_B} = \lambda_0 \frac{\partial U_B}{\partial X_B} + \lambda_2 = 0 \tag{4}$$

$$U_A(X_A, Y_A) = U_{A0} \tag{5}$$

Thus:

$$\frac{\partial U_B / \partial X_B}{\partial U_B / \partial Y_B} = \frac{\partial U_A / \partial X_A}{\partial U_A / \partial Y_A} \tag{6}$$

Equation (6) indicates $MRS^A = MRS^B$, that the marginal rates of substitution of A and B are equal. Under the case of a Pareto optimality state, any two arbitrary kinds of goods' MRS are equal. From another point of view, when A'MRS equals B'MRS, there must be a Pareto optimal state. So, the value MAX (U_B) is an optimal utility.

In the initial case, there are not resource providers and resource consumers in the system. When a resource provider appears, their resources will be quantified through the assessment system and mapped to virtual banks. When a resource consumer appears, the agent will get obtain their consumption preference and utility function. In the event that the available resources meet consumers' requirements, the system essentially obeys the principal of first-come and first-serve by directly allocating these resources. At this time, if a new kind of resource either joins the Cloud Bank or the resources were already allocated, the Cloud Bank system must carry out a refresh. Likewise, when the amount of resources available at a given time cannot satisfy the latest consumer, the system should adopt a Pareto optimal algorithm, which can maximize the utility of resources available to the last consumer without decreasing resources or utilities available to other consumers. In other words, the last consumer's interest will not be maximized at expense of any other consumer's.

8.3 The Extended Pareto Optimality Model

Consider an economy consisting of M consumers (indexed by $m = 1, \cdots, M$) and N resources (indexed by $n = 1, \cdots, N$). Consumer$m$'s preferences over consumption vectors can be described as $x_m = (x_{1m}, \cdots, x_{Nm})$ in his consumption set $X_m \subset R^N$ represented by the utility function $u_m(\cdot)$.

An economic allocation $(x_1, \cdots, x_M)$ is a specification of a consumption vector $x_m \in X_m$ for each consumer $m = 1, \ldots, M$ and the aggregate amounts various resources is available for consumption purposes represented by $(\bar{x}_1, \cdots, \bar{x}_N)$. The allocation $(x_1, \cdots, x_M)$ is feasible if $\sum_m x_{nm} \leq \bar{x}_n$[146].

Pareto optimality: A feasible allocation $(x_1, \cdots, x_M)$ is *Pareto optimal* if there is no other feasible allocation $(x'_1, \ldots, x'_M)$ such that $u_m(x'_m) \geq u_m(x_m)$ for all $m = 1, \cdots, M$ and $u_m(x'_m) > u_m(x_m)$ for some m[146].

A feasible allocation $(x'_1, \cdots, x'_m)$ that *Pareto* dominates another feasible allocation$(x_1, \cdots, x_M)$. That is, $u_m(x'_m) \geq u_m(x_m)$ for all $m = 1, \cdots, M$ and $u_m(x'_m) \geq u_m(x_m)$ foor some m[146].

If it is possible to create a Pareto optimal, we have a Pareto improvement[146].

The following section is the mathematical model of this problem.

To begin, we assume that the consumption set of every consumer is R^N_+ and that preferences are represented by utility functions $u_m(x_m)$ that are twice continuously differentiable and satisfy the gradients $\frac{\partial u_m(x_m)}{\partial x_m} \gg 0$ at all x_m(hence, preferences are strongly monotone). We also normalize, so that $u_m(0) = 0$.

The mathematical model of optimal allocation of available resources across consumers is as follows: Given some aggregate amounts $(\bar{x}_1, \cdots, \bar{x}_N)$ of resources available for consumption purposes, they must be distributed to maximize the first consumer's well-being while meeting the utility requirements $(\bar{u}_2, \cdots, \bar{u}_M)$ for consumers $2, \cdots, M$. This leaves a problem to solve,

$\mathrm{Max}_{(x_1, \cdots, x_M)} u_1(x_{11}, \cdots, x_{N1})$

a. $u_m(x_{1m}, \cdots, x_{Nm}) \geq \bar{u}_m$, $m = 2, \cdots, M$;

b. $\sum_m x_{nm} \leq \bar{x}_n$, $n=1, \cdots, N$,; (7)

c. $x_{nm} \geq 0$, $n=1, \cdots, N$, $m = 1, \cdots, M$.

8.3.1 Relative Proof

Under our assumptions, all the constraints of problem (7) will be binding at a solution.

The paragraphs below show that any allocation which is a solution to problem (7)is Pareto optimal and any Pareto optimal allocation for this economy must be a solution to problem (7) for some choice of utility levels $(\bar{u}_2, \cdots, \bar{u}_M)$.

Proof: Suppose first that a feasible allocation $(x_1, \cdots, x_M)$ is a solution to problem (7). If it is not Pareto optimal, then there exists another feasible allocation $(x'_1, \cdots, x'_m)$ where in Pareto dominates $(x_1, \cdots, x_M)$.

Because of the feasibility, $(x'_1, \cdots, x'_m)$ satisfies constraints a. and c., the Pareto dominance implies that it also satisfies constraint b.

If $u_1 x'_1 > u_1(x_1)$,

then this contradicts the hypothesis that $(x_1, \cdots, x_M)$ is a solution.

If $u_1(x'_1) = u_1(x_1)$,

then by Pareto dominance, there is a consumer $i \neq 1$, such that

$u_i(x'_i) > u_i(x_i) \geq \bar{u}_i$.

Now, by the strong monotonicity of the preferences and the continuity of the utility function, we use a small transfer from consumer i to consumer 1 to make a new allocation

$(x''_1, \ldots, x'_{i-1}, x''_i, \ldots, x'_m)$

such that $u_1(x''_1 x\ ) > u_1(x'_1) = u_1(x_1) \geq \bar{u}_i$ and $u_i(x'_i) > u_i(x''_i) \geq \bar{u}_i$, and the new allocation still satisfies the constraints (in microeconomic theory, actually, this process is a

Pareto improvement). But, this makes a contradiction to the hypo book that $(x_1, \cdots, x_M)$ is a solution.

Hence $(x_1, \cdots, x_M)$ must be Pareto optimal.

Suppose conversely that $(x_1, \cdots, x_M)$ is Pareto optimal. If $u_i = u_i(x_i)$ for each $i \neq 1$, then $(x_1, \cdots, x_M)$ is a solution to problem (A) by definition of Pareto optimality.

8.3.2 to Solve the Problem Under M×N Pareto Optimality

This thesis applied the Kuhn-Tucker theorem to solve Problem (1).

Denoted by $(\delta_2, \cdots, \delta_M) \geq 0, (\mu_1, \cdots, \mu_N) \geq 0,$ and $(v_{11}, \cdots, v_{nm}, \cdots, v_{NM}) \geq 0$ the multipliers associated with the constraints a, b and c of problem (7) respectively, with $\delta_1 = 1$.

Let

$$u_1(x_{11}, \cdots, x_{N1}) - \sum_{m=2}^{M} {}_m\left(\bar{u}_m - u_m(x_{1m}, \cdots, x_{Nm})\right)$$
$$-\sum_{n=1}^{N} {}_n\left(\sum_m x_{nm} - \bar{x}_n\right) + \sum_{nm} v_{nm} \cdot x_{nm} = 0.$$

Hence, it can obtain

$$\begin{cases} \dfrac{\partial u_1}{\partial x_{n1}} - \mu_n + v_{n1} = 0, \text{for every n;} \\ {}_m \dfrac{\partial u_m}{\partial x_{nm}} - \mu_n + v_{nm} = 0, \text{for every n and m} \neq 1 \end{cases}$$

Rearranging these terms and by eliminating the v_{nm} by the complementary slackness condition $v_{nm} \cdot x_{nm} = 0$ it obtain the first-order Kuhn-Tucker necessary conditions for problem (1):

$$x_{nm}:\ \delta m \frac{\partial u_m}{\partial x_{nm}} - \mu_n \begin{cases} = 0, & x_{nm} > 0 \\ \leq m & \end{cases} \text{For all n, m} \tag{7}$$

for

Especially, since $\delta_1 = 1$, $\frac{\partial u_m(x_m)}{\partial x_m} \gg 0$ it has $\delta_m > 0$ for all m. Thus, it can obtain one type of ratio condition (8) from (7):

$$\begin{cases} \dfrac{\frac{\partial u_m}{\partial x_{nm}}}{\frac{\partial u_m}{\partial x_{nm}}} = \dfrac{\frac{\mu_n}{\delta_m}}{\frac{\mu'_n}{\delta_m}} = \dfrac{\mu_n}{\mu'_n} \\ \dfrac{\frac{\partial u_{m'}}{\partial x_{nm'}}}{\frac{\partial u_{m'}}{\partial x_{n'm'}}} = \dfrac{\frac{\mu_n}{\delta'_m}}{\frac{\mu'_n}{\delta'_m}} = \dfrac{\mu_n}{\mu'_n} \end{cases} \Rightarrow \dfrac{\frac{\partial u_m}{\partial x_{nm}}}{\frac{\partial u_m}{\partial x_{n'm}}} = \dfrac{\frac{\partial u_{m'}}{\partial x_{nm'}}}{\frac{\partial u_{m'}}{\partial x_{n'm'}}}, \quad \text{for all } m, m', n, n' \quad , \tag{8}$$

The ratio condition (8) implies that in any Pareto optimal allocation, all consumers' marginal rates of substitution between every pair of resources must be equalized, such as the situation in the 2×2 consumption case.

Conditions (7) and (8) are equation sets for all m, m', n, n' Hence, this equation can be solved by getting to the optimal allocation $(x_{11}, \cdots, x_{N1})$ for consumer 1 and, then, the value of $u_1(x_{11}, \cdots, x_{N1})$.

We applied classical Cobb-Douglas function model as consumers' utility function that is the utility function of consumer ican be described as the following:

$u_i(x_{1i}, \cdots, x_{Ni}) = \prod_{n=1}^{N} x_{ni}^{n}$,

$\sum_{n=1\,n}^{N} = 1,\ n \in [11, 1]$

Where x_{ni} represents the number of resource n needed to consumer i, and n represents requirements weights of this resource.

Obviously, the Cobb-Douglas function is twice continuously differentiable and satisfies the gradients $\frac{\partial u_m(x_m)}{\partial x_m} \gg 0$ at all x_m.

After determining the utility function $u_i(x_{1i}, \cdots, x_{Ni})$, we can maximize consumer i's utility requirement and meet his/her most preferred resources vector, given budget and the price of every type of resource. That is to solve problem (7):

Max $u_i(x_{1i}, \cdots, x_{Ni})$

$\sum_{n=1}^{N} x_{ni} p_n \leq i$.

In problem (7), p_n represents the price of resource n, and b represents consumer i's total budget.

In solving problem (7), we can apply the Kuhn-Tucker theorem as solving problem (A). The solution to this problem is consumer i's expected utility requirement$\bar{u}_i(\bar{x}_{1i}, \cdots, \bar{x}_{4i})$, along with expected consumption vector $(\bar{x}_{1i}, \cdots, \bar{x}_{4i})$.

8.4 Cloud Banks Achieve Optimal Resources Allocation by Pareto Optimality Theory

After successfully establishing the mathematics model of Pareto optimality based on the Mix production model, we need to convert the theory into practical application. The following paragraph will discuss a resource scheduling algorithm based on a mathematical model.

The goal of resource scheduling is to achieve the highest possible system throughput and match the application needs with available system resources. Following this purpose, we designed a resource scheduling strategy for a Cloud Bank based on the Pareto optimality Mix production model.

The following scripted is the Details of the Resource Scheduling Strategy for Cloud Bank.

In the Cloud Bank, resources are divided into four types: CPU, memory, hard disk, and bandwidth. Accordingly in both problem (1) and problem (2), $N = 4$.

Each submitted task is assigned three vectors by the CCA:

Allocated resource vector:$(x_{1i}', x_{2i}', x_{3i}', x_{4i}')$, initialized as (0, 0, 0, 0) for anew task;

Expected resources vector: $(\bar{x}_{1i}, \bar{x}_{2i}, \bar{x}_{3i}, \bar{x}_{4i})$, the solution of problem (B);

Pareto optimality allocation vector: $(x_{1i}, x_{2i}, x_{3i}, x_{4i})$,the solution of problem (A).

After users submit tasks, the CCA analytic and estimates them based on classifying consumers' service requirements, which gives corresponding weights vector $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. However, being neither analysed by the consumers nor by the Cloud Bank system, this process cannot exactly match reality. So, we need the CCA to refresh the consumers' utility functions by cycle and dynamically adjust every weights vector $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ according to the operation of tasks.

The resources scheduling strategy shown as follows:

Step1: The CCA distinguishes whether consumer i submits specific needs directly, if true go to step2, else go to step3;

Step2: For everyn, x_{ni} is assigned to what the consumer submitted.

Step3: The CCA analyses and estimates a given task's level, and then generates the utility function $u_i(x_{1i}, \cdots, x_{4i})$.

Step4: In Solving Problem (2), obtain$\bar{u}_i(\bar{x}_{1i}, \cdots, \bar{x}_{4i})$;

Step5: Comparing $\bar{x}_{ni}$ and x'_{ni} for every n, if there is n such that $\bar{x}_{ni} < x'_{ni}$ then go to step6, if there is n such that $\bar{x}_{ni} > x'_{ni}$ then go to step7, if$\bar{x}_{ni} = x'_{ni}$ for every n then keeps running task and waits CCA Updating.

Step6: Global Allocator puts these extra resources from consumer back to resource pool;

Step7: Solving problem (1), obtain $u_i(x_{1i}, \cdots, x_{4i})$;

Step8: Comparing x_{ni} and $\bar{x}_{ni}$ for every n, if there is $x_{ni} \geq \bar{x}_{ni}$ for every n then go to step 9, else go to step10;

Step9: Let $\left(x'_{1i}, x'_{2i}, x'_{3i}, x'_{4i}\right) = (\bar{x}_{1i}, \cdots, \bar{x}_{4i})$;

Step10: If there are not enough resources in the Cloud Bank at the moment, the consumer awaits the CCA refresh to request the needed resources. If the task is not assigned to resources after three cycles, the Global Allocator forces it to allocate resources by latest resources requirement.

In the strategy above, the CCA refreshes user's utility function by this cycle since this process is a continuous loop.

8.5 Improvement of PO-based Allocation Strategy

There are still some issues to discuss in this strategy, as follows:

Only grading of consumers' tasks and classifying those services which consumers requested exactly as much as possible, so achieving the SLA between the Cloud Bank and consumers would be guaranteed, and the CCA generates weights vector $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ more in line with actual needs of consumers. Secondly, the possibility of consumers submitting specific resources to the CCA directly is quite low. For instance, an end user, such as a small shop owner, likely could not know how many specific resources he/she needs, and what he/she should do is to submit a service request called "statistics" to the Cloud Bank and then let the system solving this problem.

The CCA is able to refresh the utility function in two ways, centralized and autonomous. The centralized method allows the utility functions' refreshed-cycle determined uniformly according to the system clock. For the autonomous method, because of consumers entering and quitting the Cloud Bank periodically, the CCA could refresh the utility functions according to each consumer's clock.

Similarly, it's worth considering how the CCA places those extra resources back. First, it can put them back into the resource pool directly for the convenience of statistics for the risk of Cloud Bank. Most of the time, the scheduling system exists in the state of Pareto improvement. When the number of users reaching saturation, it maybe achieve Pareto optimality. From the partial perspective, however, those allocated resources and consumers are in a Pareto optimal state.

Based on the above study, this thesis adds QoS Restraints to a PO-based Resource Allocation Strategy. To guarantee a minimum quality of service (QoS), infrastructure service providers must to enter into a legal agreement. Typically, the QoS parameters are related to the availability of the CPU, memory, hard disk, and network needed for efficient execution of the application at peak loads. This legal agreement is known as the infrastructure service-level agreement (SLA)[147].

Once the infrastructure SLA is established, it needs to be validated and distributed to some parts of the cloud related to measure the runtime parameters of cloud providers' resource, evaluate the service-level objectives (SLO), take corrective actions on violation of the SLO [148], etc.

Given the relatively new nature of Cloud Computing, the traditional usage of SLA is not fully integrated into cloud systems, especially not in regards to distributing resources. To better meet the QoS, we need to deploy an infrastructure SLA to the resource scheduling step.

Producers use a benchmark application from the Cloud Bank to grade of their resources (VM) after being virtualized. The results of this benchmark will be immediately sent to the Cloud Bank. VMs that have different performances correspond to different VM ability level, which is described by a VM ability level.

Consumer's needs/task also have to be graded to correspond with infrastructure resources. Grading to tasks is not only done via traditional IT methods but also by using a statistical approach. By collecting data regarding consumers' requirements, budget, and even loyalty to this Cloud system and then analysing the information and then pricing appropriately, the

Cloud Bank is able to guarantee to every consumer that the infrastructure SLA (the value of VM ability level) must less than or equal to the expected utility requirement$\bar{u}_i(\bar{x}_{1i}, \cdots, \bar{x}_{Ni})$. Finally, an SLO can be added into the infrastructure SLA document to describe that service or resources of a certain level requirement must use a minimum of certain VM ability level.

8.6 The steps of dynamic simulation

In order to verify the effectiveness of resource scheduling algorithms, we have a secondary development based on the CloudSim API. This section is based on CloudSim API to add resource scheduling algorithm to achieve the desired type of cloud computing resource scheduler. Additionally, it also lays the foundation for the experimental analysis. The resource scheduler envisioned to accomplish the needed duties must include several features: the ability to build the queues of the users need tasks; the ability to dynamically discover resources and according to the numbers of participants to dynamic equilibrium resource price; the ability to determine the number of participants based on the resource dynamic equilibrium resource price; the ability to map out a specified task to the resource based on the algorithm; the ability for tasks to be submitted to the appropriate resource node and return the results to the user; and finally the ability to demonstrate the results of task execution. According to the above functional description, the resource scheduler in the structural design of several major classes shows in Fig.8.3

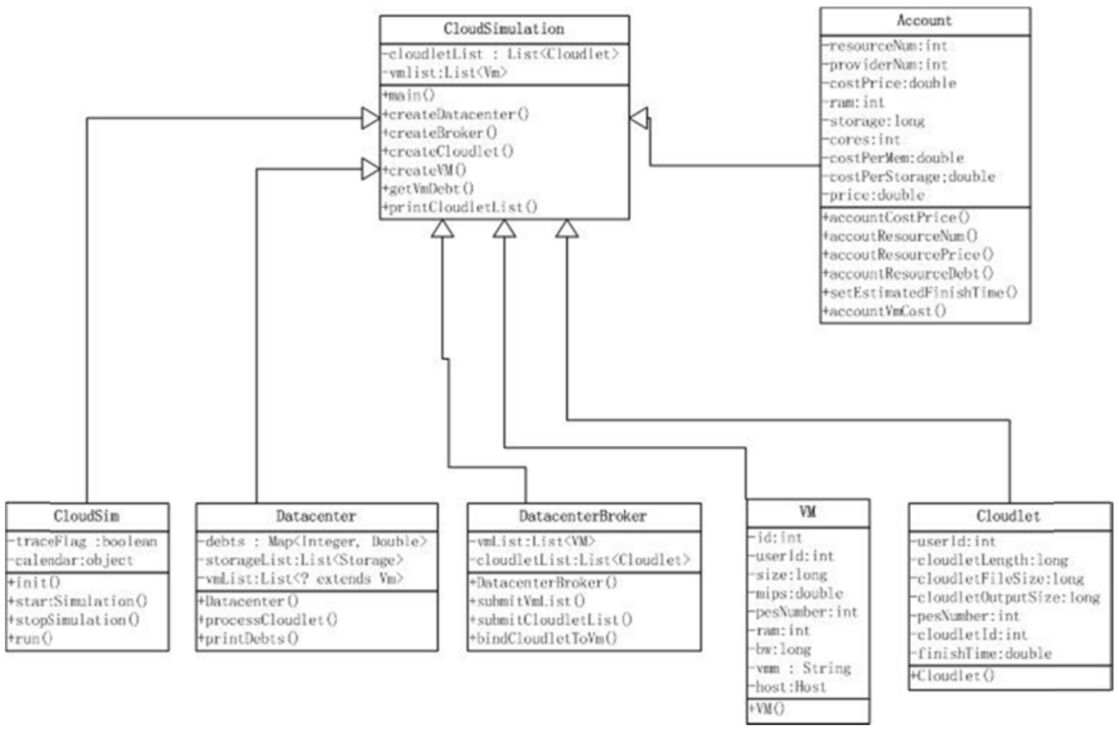


Figure 8. 3 Resource Scheduling Class Structure Diagram

These only list the various classes and methods during the simulation. The CloudSim provides the main roles of CloudSim, Data Centre, Datacentre Broker, VM, Cloudlet, etc. The design implements the resource balancing strategy and resource scheduling algorithms separated into Account and Cloud Simulation classes. Both of these support the expansion of the algorithm.

The relations among those entities and the sequence diagram are shown below:

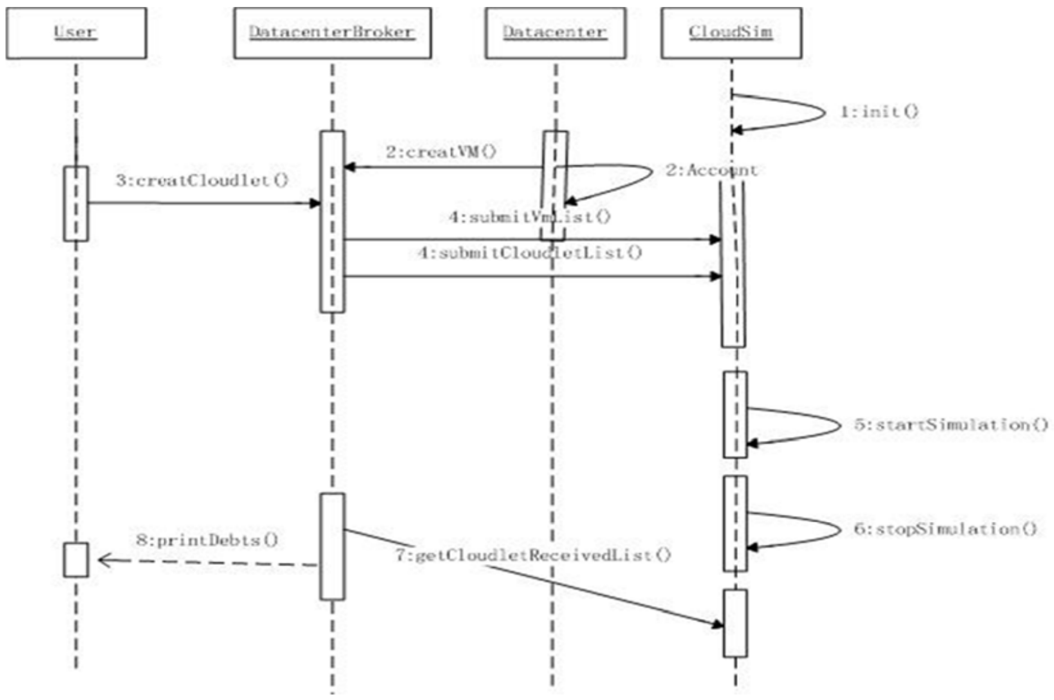


Figure 8. 4 the Scheduling Sequence Diagram

The figure describes resource scheduling process. The process of step 2 is to create VM (resources) and presents all the available resource when the resource providers dynamically join and exit the cloud. Through step 7, we can obtain the user's task detailed information about the task id, provider's id, and the time cost. In step 8, we can determine the total cost about the process the user's task.

8.7 The Simulation Environment Set Up

Our Cloud Bank was simulated using the following parameters. First, the installation and configuration of the CloudSim on the environment is detailed in the following: Windows XP or newer. Eclipse version: Indigo Service Release 1. JDK 1.6, CloudSim requires JDK version newer than 1.6 (which can be downloaded from: http://www.oracle.com/technetwork/java/javase/downloads/index.html), though the environmental variables needed to be configured for JDK. Second, CloudSim 2.1.1 was downloaded (http://www.cloudbus.org/cloudsim/doc/api/index.html) and then simply unpacked to complete the installation. To remove CloudSim, the directory can simply be removed.

The directory structure of CloudSim is as follow:

```
cloudsim/        -- top level CloudSim directory
docs/        -- CloudSim API Documentation
examples/        -- CloudSim examples
jars/        -- CloudSim jar archives
sources/        -- CloudSim source code
tests/        -- CloudSim unit tests
```

The JAR files are provided to compile and to run CloudSim applications:

* jars/cloudsim-<VERSION>.jar   -- contains the CloudSim class files

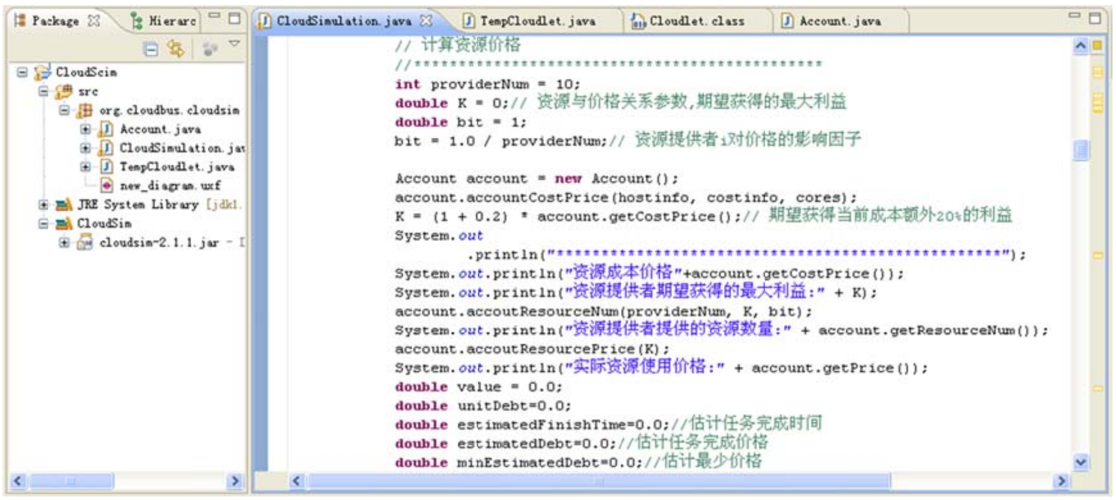
* jars/cloudsim-<VERSION>-sources.jar-- contains the CloudSim source code files

* jars/cloudsim-examples-<VERSION>.jar -- contains the CloudSim examples class files

*jars/cloudsim-examples-<VERSION>-sources.jar -- contains the CloudSim examples source code files

Third, as CloudSim development kit is based on Java, it is advantageous to choose the most stable development environment Eclipse. The following specific description of the experimental environment includes: Windows XP, Memory of 1G,CPU, 1.6GHz Dual-core;JDK1.6; CloudSim-2.1.1; Eclipse 3.5.

The process was simple, first create a java project in Eclipse, import the CloudSim Development Kit, then create and implement the resource balancing strategy and resource scheduling algorithms as separate Account and CloudSimulation classes. The IDE is as shown below:



```
// 计算资源价格
//*********************************************
int providerNum = 10;
double K = 0;// 资源与价格关系参数,期望获得的最大利益
double bit = 1;
bit = 1.0 / providerNum;// 资源提供者i对价格的影响因子

Account account = new Account();
account.accountCostPrice(hostinfo, costinfo, cores);
K = (1 + 0.2) * account.getCostPrice();// 期望获得当前成本额外20%的利益
System.out
        .println("*******************************************************");
System.out.println("资源成本价格"+account.getCostPrice());
System.out.println("资源提供者期望获得的最大利益:" + K);
account.accoutResourceNum(providerNum, K, bit);
System.out.println("资源提供者提供的资源数量:" + account.getResourceNum());
account.accoutResourcePrice(K);
System.out.println("实际资源使用价格:" + account.getPrice());
double value = 0.0;
double unitDebt=0.0;
double estimatedFinishTime=0.0;//估计任务完成时间
double estimatedDebt=0.0;//估计任务完成价格
double minEstimatedDebt=0.0;//估计最少价格
```

Figure 8. 5 Project Environment

8.8 Running the CloudSim Instance

Based on the simulation platform and resource scheduler, this section detail show we simulated the resource scheduling in the cloud computing environment. The specific simulation procedures are as follows:

Initialize the simulation program, the main initialization information are the current number of users, the system clock and other parameters, then registration user’s information into the CloudSim.

Step 1: Initialization

```
        int num_user = 1; // Number of users
        Calendar calendar = Calendar.getInstance();//System clock
        boolean trace_flag = false;
   CloudSim.init(num_user, calendar, trace_flag);//Initialization CloudSim package
```

Step 2: Creating Data Center

The data centreconsists of resource providers. The host in the data centre is the amount of resources offered by the resource providers. One host can be composed by multiple VMs (virtual machine) according to the description of parameters. A separate virtual machine performs user tasks.

```
        //creating data centre, which is representative of each resource provider
   Map hostinfo = new HashMap();//Storage hostdescriptioninformation,
                                          includingmemorysize,diskspace, etc.
```

```
Map costinfo = new HashMap();//Storageunitprice ofresource use, including
                              CPU, storage, memory,ect.
int hostnum =1;// number of resource
int cores = 1;// number of CPU
Datacenter datacenter0 = createDatacenter("Datacenter_0", hostinfo,
costinfo, hostnum, cores);
```

Step 3: Creating task scheduler

The taskscheduleris responsible forreceivinguserrequestsand also registration informationresources. At the same time, aftermatchingthe scheduling algorithm, the task schedular alsobinds resources to perform specifictasks.

```
//creating broker which responsible for the registration list of virtual machines and task
   list.
         DatacenterBroker broker = createBroker();
         int brokerId = broker.getId();
```

Step 4: Creating VM and the list of user's tasks

Whenthe user submitsa task, includinga description ofsome specificparameterssuch as tasklength,file size, the number of processorsrequired, etc., the VMsimulateuser taskssubmission via the method of "creatCloudlet" which createsa list of the user's tasks.

```
         // Creating virtual machinelist andtask list;
         vmlist = createVM(brokerId,10); // Creating 20 virtual machines
cloudletList = createCloudlet(brokerId,9,tempCloudlet); // Creating 40 tasks
```

Step 5: Submission of the task and the current resources to CloudSim

When the user's tasks list and resource initialization is complete, those two lists are submitted into the CloudSim.

```
//Submitthe list ofvirtualmachines and task listto thebroker
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);
```

Step 6: Perform the scheduling algorithms and allocate resources to perform the tasks.

According the current the task information and the description of resource,the scheduler pre-estimates the execution time for each task based on the resource provider. Through the matching algorithms, the schedulingchooses the shortest time with minimum cost to complete the user's resources, binding those tasks and resources together and then submitting them to the CloudSim to perform the tasks.

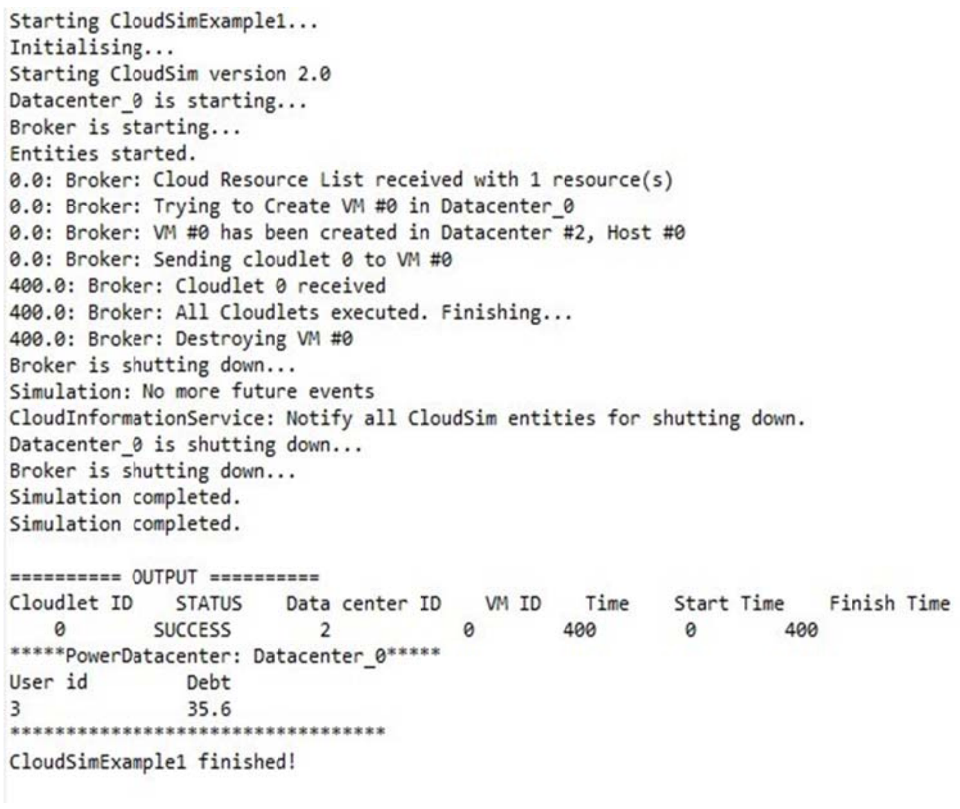
```
        tempaccount.setEstimatedFinishTime(cloudlet.getCloudletLength(),
        vm.getMips(), vm.getPesNumber(), cloudlet.getPesNumber());
value=tempaccount.accountVmCost(vm,cloudlet,costinfo,hostinfo,
       cloudlet.getPesNumber());
        account.accountResourceDebt(value);
        unitDebt=account.getUserDebt();
        broker.bindCloudletToVm(cloudlet.getCloudletId(),i);
        CloudSim.startSimulation();
```

Step 7: Return the results

The scheduler broker obtainsthe task execution results, which includes the task execution time, cost, mission number and other information about the resource providers, which are then returned to the user.

```
        //output the result
        List<Cloudlet>newList = broker.getCloudletReceivedList();
        printCloudletList(newList);
        datacenter().printDebts();
```

As the simulation program is quit long, full code listed in Appendix.



```
Starting CloudSimExample1...
Initialising...
Starting CloudSim version 2.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: VM #0 has been created in Datacenter #2, Host #0
0.0: Broker: Sending cloudlet 0 to VM #0
400.0: Broker: Cloudlet 0 received
400.0: Broker: All Cloudlets executed. Finishing...
400.0: Broker: Destroying VM #0
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.

========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2            0        400        0        400
*****PowerDatacenter: Datacenter_0*****
User id          Debt
3                35.6
**********************************
CloudSimExample1 finished!
```

Figure 8. 6 Result of run CloudsimExample1

8.9 Analysis of Experimental Results

The experiments this thesis compared between the min-min resource scheduling algorithms and the price mechanism of scheduling algorithm in minimum completion time in order to compare the efficiency of resources and the cost of the user.

During the program initialization period, the default is to have no resource providers. As the provider's resources continue to join, the amount of provider's resources continues to change. The parameters of the provider's resource is detailed in Table 8.1

Table 8. 1 Parameter List of Resources

Resource Provider	Host No.	Quantity of the CPU	Each CPU processing speed (MIPS)	Memory (Mb)	Hard Disk (Mb)
D0	D00	1	400	1000	160000
	D01	2	200	2000	200000
D1	D10	2	300	2000	100000
	D11	4	500	4000	900000
	D12	1	250	2000	400000
D2	D20	2	900	8000	800000
	D21	4	800	6000	500000
	D22	3	1000	5000	600000
	D23	2	700	7000	400000

Suppose the user has 100 tasks, the length of each task randomly between the 5000MIP to 50000MIP. The components resource cost are known, the CPU time as the unit price of 3 dollar / each, disk unit of time 0.001 dollar/G, memory 0.05 dollar /G, etc. Observe the same number of tasks in the two algorithms under the premise of implementation of mandates. The task of estimating the minimum execution time is shown in Table 8.2.This process gives the load balancing execution time and the user cost of state in the minimal execution time, yielding the results shown in Figs.8.7, and 8.8.

Table 8. 2 List of Task Execution Time on Each Resource

Resource Task	R0(s)	R1(s)	R2(s)	R3(s)	R4(s)	R5(s)
T0	200	500	150	320	170	420
T1	310	600	160	360	230	500
T2	400	700	280	330	320	640
…						

The resource scheduling load balancing is an important objective of the cloud computing, which directly affect the overall efficiency of resource management system. The following experiment is shown the improved min-min algorithm can significantly improve resource utilization.

Using the min-min algorithm to perform the task and resource, the Fig.8.7 shows that the resource R2 to implement is the fastest. All tasks are performed on that resource, even in the waiting queue. There were, however, other free resources available, leading to a serious imbalance in which all tasks are to be performed on R2. The execution time of the task also includes the actual execution time plus the waiting time, and accordingly the actual execution time is significantly longer.

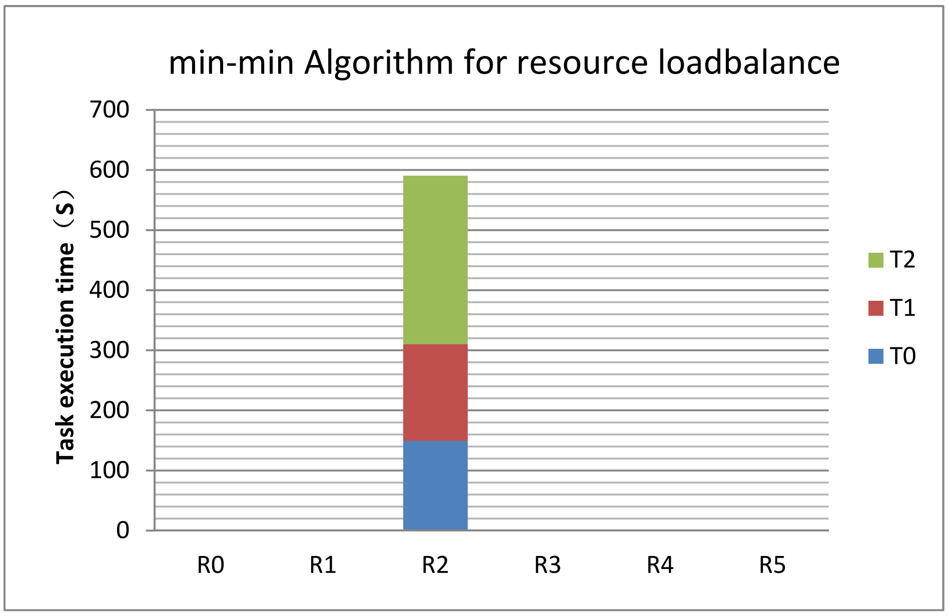


Figure 8. 7 min-min Algorithm for Resource Load Balance

Using the price mechanism based improved min-min algorithm, all the tasks were not performed on the fastest resource, but were matched with sub-optimal resources. This process will not create a load imbalance problem by reducing the waiting time of a task and significantly improving the utilization of various resources. The results of this experiment are shows as Fig. 8.8.

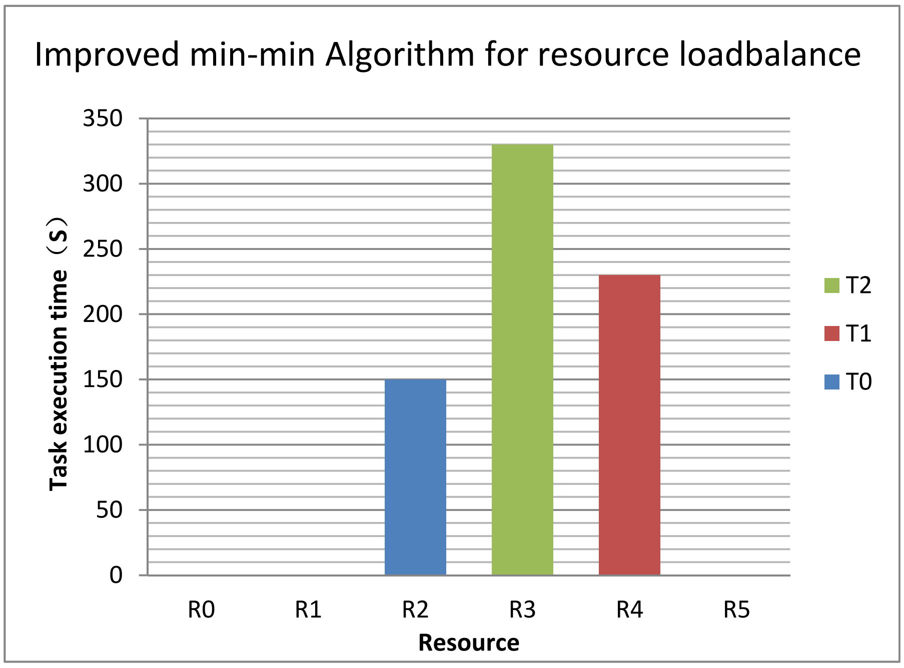


Figure 8. 8 Price Mechanisms Based improved min-min Algorithm for Resource Load balance

In addition to meeting the load balancing of resources, it not only improves resource utilization, but also meets the requirement of the consumer with minimal cost to completing the task in less time. The following cost comparison experiment verified that the adoption of the improved min-min algorithm allows completing the task in less time and significantly reduced user costs. The min-min algorithm will make the tasks wait a comparatively long time in the queue with the same resources, thus increasing the actual execution time of task. Fig. 8.8 shows that the min-min task execution time of the algorithm is generally higher than to the improved min-min algorithm with the steady increase in the number of tasks that can be seen in Fig. 8.9 shows the introduction of the price mechanism algorithm leads to the user cost significantly being lowered as to the min-min algorithms. This experiment ignores the random delay of the network, file transfer time, node bandwidth, algorithm cost, and load conditions, which may not be a true reflection of the curve, but the implementation should also be able to largely react to the task being performed.

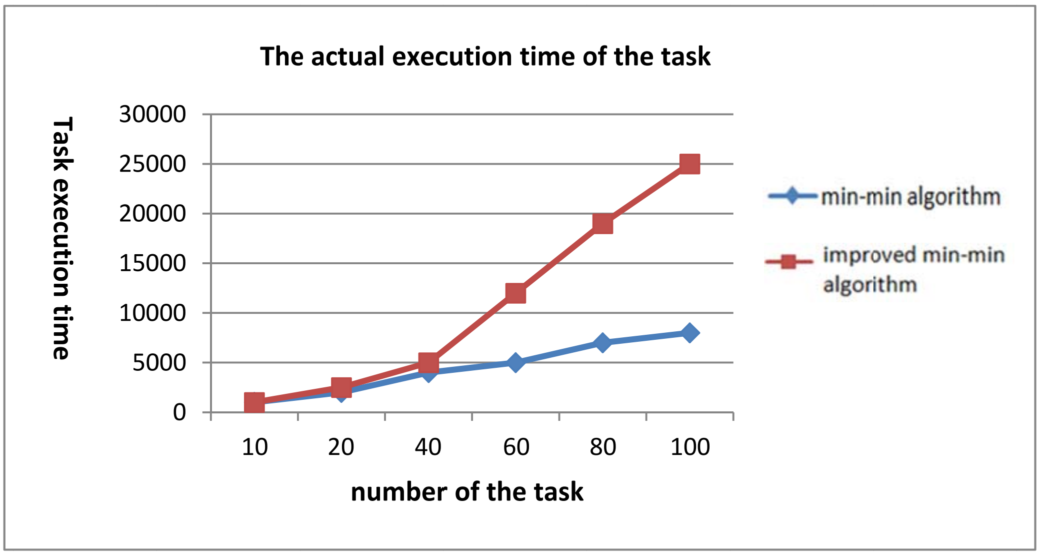


Figure 8. 9 Comparison of actual execution time task

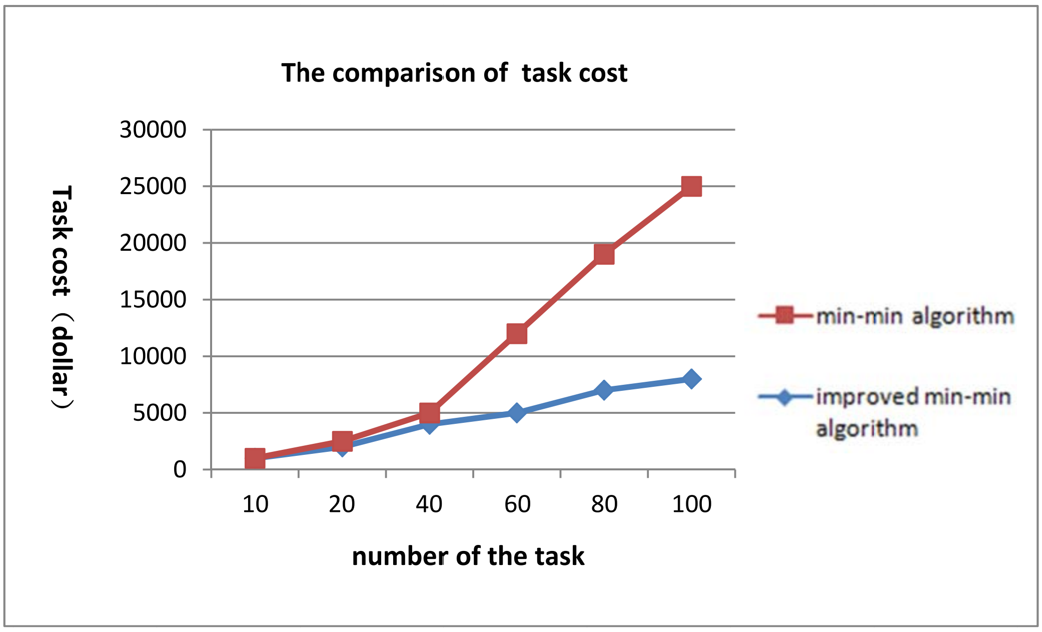


Figure 8. 10 Comparison of task cost

8.10 Summary

In the problems of resource scheduling and allocation, “fairness” is hard to define and even more difficult to achieve. It is of little wonder that Pareto optimality is called the “Ideal kingdom” of fairness and efficiency. Pareto improvement can effectively achieve resource scheduling and allocation under the condition wherein the system has to match consumers’ requests (for our purposes, namely, utility function). This observation has been proved in both social and economic fields. Accordingly Pareto optimality has been referred to as Pareto efficiency.

Traditionally, in distributed computing, solving the problem of resource scheduling is only approached from point of pure computer science that ignores the essential “resource.” In the field of economics and social sciences, achieving a fair and effective resource allocation among individuals and groups has been explored for perpetually throughout human history. Pareto optimality from microeconomics has been applied in industrial production and mechanical design since the appearance of the Industrial Revolution. In recent years, economic theories have increasingly found application in computer sciences.

Similar to the rise in the application of economic theory to first production and design and then later to computer science, our Cloud Bank is established using many aspects of a commercial bank model. Similarly, transaction and distributing infrastructure resources are very similar to the social resources assigned among social groups and individuals. Hence, we can apply the Pareto optimality theory to simulate the resource scheduling process in cloud computing, and an actual formal description can be easily achieved by programming.

Despite the usefulness of borrowing from commercial banking in developing our Cloud Bank as well as using economic theory to improve transactions, there are some key differences between cloud computing and practical economics. For starters, to guarantee the effectiveness of achieving resource scheduling in cloud computing (or QoS), people apply SLA but not laws—true, SLAs are bound by some legal frameworks, but the specifics are largely negotiated between the parties using the Cloud Bank, and not by outside entities. Various SLO in SLA describe rights and obligations of resource providers, resource consumers and even the Cloud Bank itself.

In order to verify the validity of the Cournot equilibrium pricing strategies as well as task scheduling algorithms, this chapter examined the experimental analysis of the proposed algorithm. This chapter first analyzed the cloud computing simulation tools, CloudSim, regarding the detailed modules and functions, and secondly, introduced the detailed description of the resource scheduling process based on CloudSim. Finally, we discussed one of the improved algorithms mentioned and how it was simulated in the CloudSim. Finally, this chapter verified that the improved resource scheduling algorithm for load balancing would likely improve effectiveness and user satisfaction, at least under simulated circumstances.

Chapter 9 **The Real Laboratory Platform: IaaS based cloud computing platform**

This chapter presents the real cloud computing platform, the IaaS category. Using this platform offers a simulated environment in which to experiment as to some of the theories behind the Cloud Bank. All related lab work has been separately presented in each chapter, but due to the limitations inherent in presenting such a project in the limited scope of a dissertation, , some of code been moved into the appendix.

9.1 Introduction

IaaS (Infrastructure as a Service) provides data centres, infrastructure hardware, and software resources through the Internet. More than that, IaaS provides server, operating system, and disk storage, database/information resources. Among the major players, the most representative of IaaS products are part of Amazon AWS (Elastic Compute Cloud), but IBM, VMware, HP and some other well-known IT companies also provide these types of service. IaaS usually bring into other and use mode and pricing mode according to "elastic cloud" mode that is using the service what you need and paying for it in any particular time. The contents are shown as below:

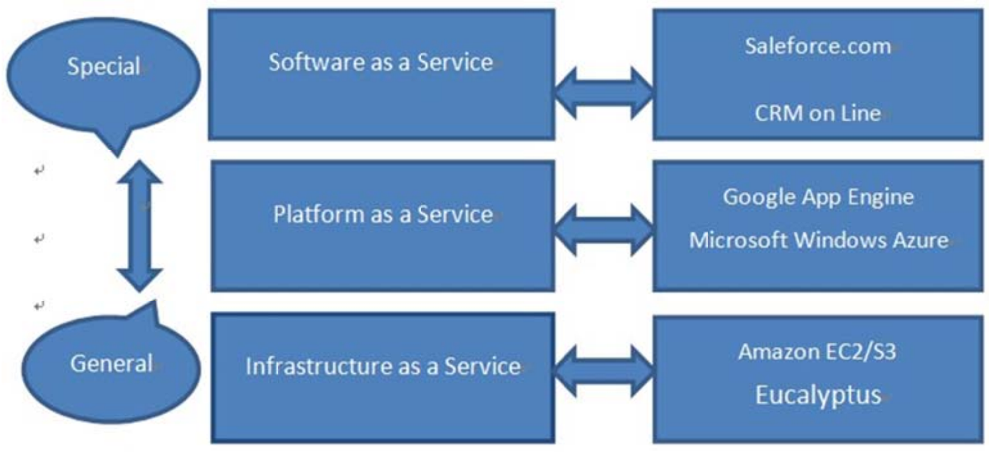


Figure 9. 1 Three Types of Cloud Computing

At present, the most representative application for cloud computer is Google cloud, which includes a variety of both enterprise and personal applications including GoogleDocs, GoogleApps, Googlesites, as well as host of other services still under development in GoogleLabs. One Google documents are word processing and spread sheet program that are based on the network. It can improve the efficiency of cooperation. Users can update the files on-line at the same time and can real-time see other members' edition[149].

9.2 Setting up the IaaS based cloud computing environment

9.2.1 The comparison of the two kinds of the platform structures

At present, in the open source environment, there are two options for platform structures: OpenStack or Eucalyptus. OpenStack is a global collaboration of developers and cloud computing technologists producing the ubiquitous open source cloud computing platform for public and private clouds. The project aims to deliver solutions for all types of clouds by being simple to implement, massively scalable, and feature rich. The technology consists of a series of interrelated projects delivering various components for a cloud infrastructure solution. Founded by Rackspace Hosting and NASA, OpenStack has grown to be a global software community of developers collaborating on a standard and massively scalable open source cloud operating system. All of the code for OpenStack is freely available under the Apache 2.0 license. Anyone can run it, build on it, or submit changes back to the project.

Eucalyptus is another building plan. One of the great advantages of Eucalyptus is its own open source software components can be used without modification. It also means it can easily run in GNU Linux kernel without modification. Ubuntu's embedded cloud computing platform is also based on Eucalyptus. It can be installed after download and its operation is very convenient.

9.2.2 Introduction of Eucalyptus

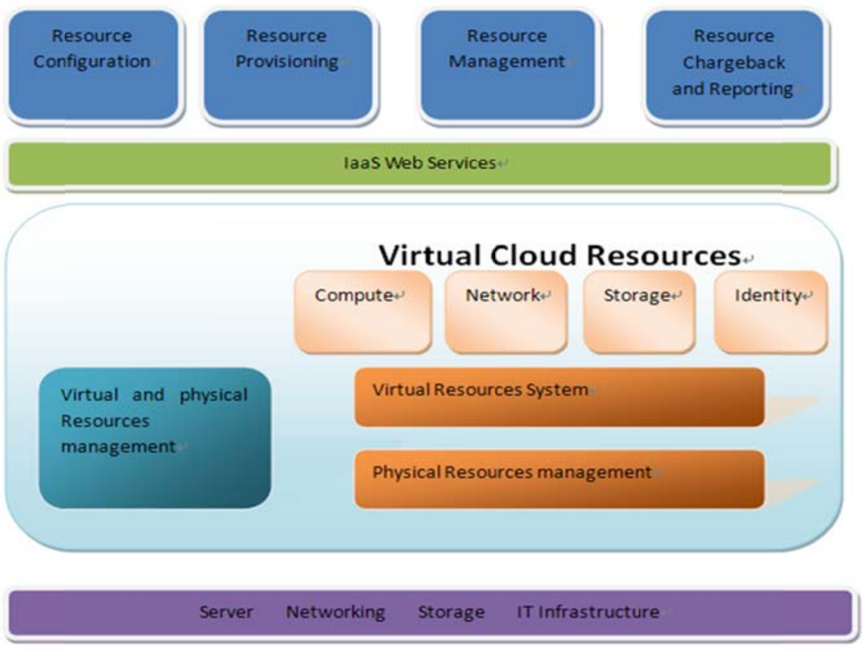


Figure 9. 2 The Infrastructure Layer Overall Function Structure Diagram[150]

The IaaS based cloud computing environment uses a virtualized CPU to install basic platform cloud Eucalyptus 1.6.2. The Eucalyptus project , whose full name is Elastic Computing Architecture for Linking Your Program to Useful Systems (EUCALYPTUS), is a kind of open source software infrastructure used to make flexible, practical computing clouds through the calculation of cluster or workstation. Originally a research project that belongs to University of California Santa Barbara Computer Science College, it has now been commercialized and become the Eucalyptus Systems, Inc. However, Eucalyptus still maintains and develops work according to the open source parameters as Eucalyptus Systems is based on the open source of Eucalyptus to construct additional products. The corporate arm also provides support services. Today, EUCALYPTUS can be installed on most of modern Linux distributions quite easily. The system provides the following advanced features:

1. EC2 and S3 interface compatibility (SOAP interface and REST interface). All existing tools that use these interfaces to make cooperation with cloud based on Eucalyptus.
2. Support running in Xen hypervisor or VM operation above KVM. The future version is also expected to support other VM types, such as VMware.
3. Used to carry out system management and users settlement cloud management.

4. They can put many groups that own respective private internal network address to configure to a cloud cluster.

9.2.3 EUCALYPTUS platform advantage

EUCALYPTUS has no special hardware demands for an enterprise data centre, as it can be deployed with a mix public, enterprise and private clouds. In current IT infrastructure, through using Linux and web services technologies, EUCALYPTUS allows clients to easily and quickly create computing clouds suitable for their particular application demands. At the same time, EUCALYPTUS supports the popular AWS cloud interface from Amazon. By using a general programming interface, these private cloud and public clouds can freely interact with one another. Along with the development of virtual machines, EUCALYPTUS already supports a cloud environment and safe network storage virtualization. The advantages of this model cannot be overstated, but in brief there are several items worth noting:

- EUCALYPTUS can safely virtualized server, network and storage, in order to reduce costs, improve maintenance convenience, and provide user self-service;
- The modular design of the EUCALYPTUS allows for a wide range of users (administrator, developers, and managers, hosting customers). Different user interfaces and virtualization technology likewise enhance this benefit, and for service providers EUCALYPTUS provides a consumer pricing model based on the operation platform;
- Both VM and cloud snapshot raise high-reliability of cluster ascension, template operation and automation, which together makes clouds based on this system easy to use by reducing the average learning curve and shortening project cycles;
- EUCALYPTUS makes full use of the existing virtualization technology based on Linux and supports a variety of different management programs;
- Convenient cluster and usability area management allow for the design of customized logic servers, storage and network for each project;
- The core of the EUCALYPTUS framework will continue to remain open source, allowing for perpetual development and improvement from a global community;

- Working to development of the public cloud compatible interface is a unique advantage, as future users will have access tp their own private clouds, allowing the public-private cloud mixtures;
- Amazon is forming a core area of its business around the AWS, and is rapidly developing this technical system. For example, RightScale, CohesiveFT, Zmanda, rpath, etc., are based on EUCALYPTUS using Amazon AWS for solutions To potential problems;
- EUCALYPTUS incompatible with many Linux distributions, including Ubuntu, hatred Hat, openSUSE, Debian, Fedora, and CentOS. Likewise EUCALYPTUS is developing management procedures and all kinds of virtualization technology, which use the License FreeBSD, meaning that it can be directly used in commercial software application. The current support business service, Amazon EC2, will continue to add a variety of client interfaces;
- The system USES makes maintenance very convenient. Using internal communication in the SOAP safe, the main design goal uses retractable systems, thereby imparting some characteristics that are easy to use and expand upon.

9.2.4 EUCALYPTUS Platform Framework

At the beginning of the EUCALYPTUS design, we ensured safety and ease of installation as far as was possible with the extant technology. The software framework consists of a series of highly modular coordination web services (which means it uses standard communication protocols giving an interactive operation). Through this framework, EUCALYPTUS realized the virtual machine and storage resources. These resources consistent of an isolated 2 layer network connection. From the client application and the user's point of view, other interface can be customized, as the EUCALYPTUS API is compatible with the Amazon AWS (including SOAP and the interface support).

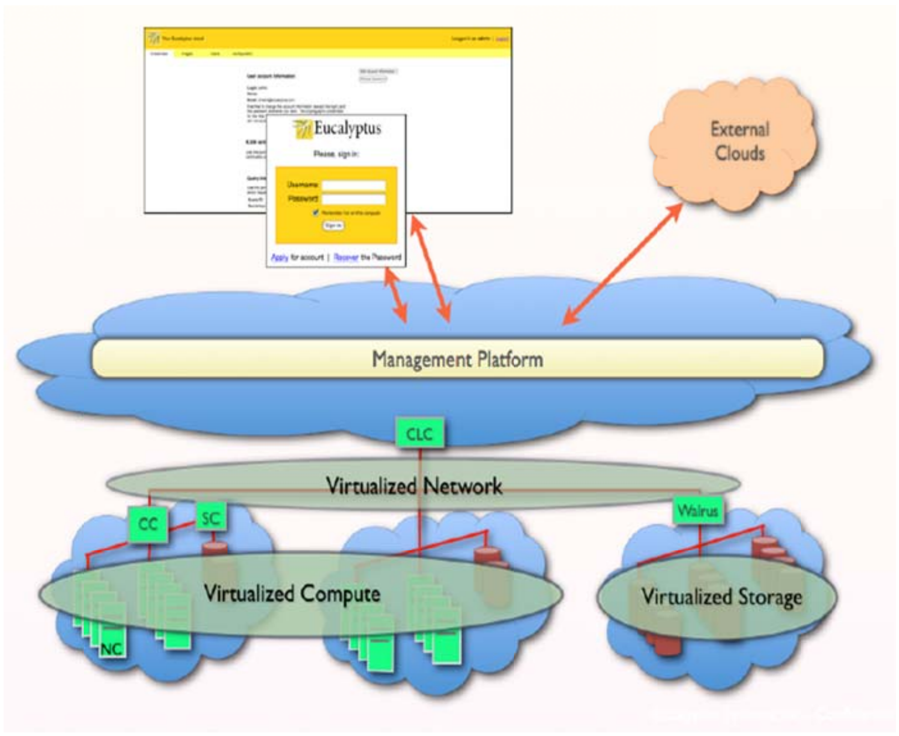


Figure 9. 3 EUCALYPTUS Cloud Concept Map[151]

9.2.5 Eucalyptus Components

Each EUCALYPTUS service kit describes a clear API. In these document descriptions, WSDL includes the service which provides operations for the input/output data structure. Internal authentication is accomplished through the standard of service-Security system WS. Overall, the installation of EUCALYPTUS includes five levels, each with its own Web Service interface (Fig.8.3). The individual EUCALYPTUS system components are as follows:

- Cloud controller (CLC), the entrance that administrators, developers, and project managers as well as end users see in cloud platform. CLC is responsible for the management of the query node of the information resources, and making the top plan. Through the Cluster controller (CC), the CLC makes a request to finish a given project (Shown as Fig.8.1). It is also the management platform CLC connect an interface. In essence, through a clear industry standard API (Amazon EC2) and a web based user interface the CLC is responsible providing a graphic user interface to manage resources (i.e., server virtualization, network, and storage).

- Cluster controller (CC) often runs in the front of the cluster machine or any one of the connections of the node controllers (NC) that can run the server or the CLC machine. CCs collect a series virtual machine in specific information and run the virtual machine. The CC also manages the network and provides a virtual example with which to participate in the management of CLC SLAs. A CC the following all nodes must be in the same broadcast domain (Ethernet).
- Node controllers (NC) provide a virtual machine running in the service of every node. The NCs control the virtual machine operation, inspection, examples of stop, extract and remove mirror (kernel, root file system and ram disk mirror) local copy, inquires and the control system software.
- Storage controllers (SC) realize the network storage of access (such as Amazon elastic piece of EBS) and storage-with many storage systems (NFS, iSCSI etc) connections. Elastic pieces of storage is a Linux piece of equipment that can be connected to the virtual machine and send data to the local mapping networks, thus acting as a remote storage location. On the whole, SCs send local connection traffic network to a remote location disk. Though through roll can't cross case sharing EBS, but allowed to create, store in central storage systems, such as Walrus service.

Walrus (get) any/storage containers and objects based on consistency, allows users to store data persistence and allows the user to create or delete, containers, submit a list, acquisition, and set the access control strategy. The Walrus interface and Amazon's S3 compatible support the Amazon Machine Image (AMI) graphics management interface, providing the ability to store and access both virtual machine mirrors and the mechanisms of user data.

Management platforms for all kinds of EUCALYPTUS services and modules provide an interface, which can include virtual machine management, storage management, the user/group management, accounting, management, the SLA definition and implementation, cloud expanded, supply, etc.

The CLC cloud controller is the foundation of the virtualization of resources (servers, storage and network). The Cluster controller (CC) is defined in the cloud of the front of each cluster. NCs run the virtual machine examples of machines while the storage controller (SC) provides pieces of storage (similar to Amazon service and through EBS). Walrus extended to the whole cloud storage system is similar to the function of libraries in the S3. On the whole, the system provides a one-stop cloud administrator console to allocating and managing the cloud. Management platforms provide various privilege based interfaces for the administrator, the project manager, the developers as well as other users[152].

9.2.6 EUCALYPTUS Configuration

Through these components, EUCALYPTUS can be configured to manage many infrastructure functions and topology structure, such as four different network models. The administrator can then adjust the cloud platform according to the corresponding security protection level in order to meet the local security configuration strategy and management needs. It also can be in a different management program and contains the unity of the virtual technology platform and work out unified deployment of EUCALYPTUS API. So, the EUCALYPTUS cloud can act in a variety of roles or technologies (used for the particular data centre need during each life cycle) while remaining unified in a single platform.

A EUCALYPTUS cloud installation can polymerize and management one or many clusters. A cluster is connected to the same LAN or group of machines. In a cluster, there can be one or many NC examples, and each instance there are examples of the instantiation of virtual management and termination.

A single cluster installation (Fig. 8.4) will include at least two machines: one machine running CC, SC and CLC and another machine running as a NC. This configuration is suitable for the purpose of the Cloud Bank experiment, given its rapid and easily customized configuration.

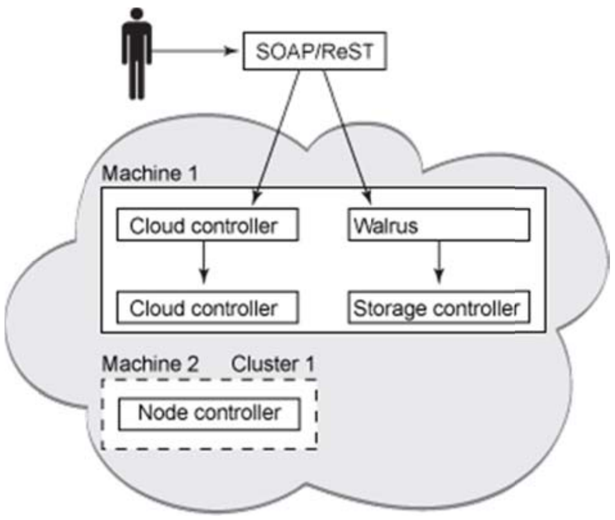


Figure 9. 4 Sole Cluster Eucalyptus Installation Topology[152]

In many clusters, each component installation (CC, SC, NC and CLC) can be placed in separate machine. If one wanted to use it to execute major task, a particular kind of configuration EUCALYPTUS cloud would be ideal. Many clusters of the installation could choose to run on through its controller type adaptation of the machine in order to significantly improve performance. For example, we can choose a computer that has a super CPU to run as CLC. Many of the clusters are involved in the improvement of the usability, load, and resources across a cluster distribution. The cluster concept is similar to the availability of Amazon EC2. Resources can cross many regional distributions, so that one faulty area will not affect the entire application. Figure 9.5 provides an example.

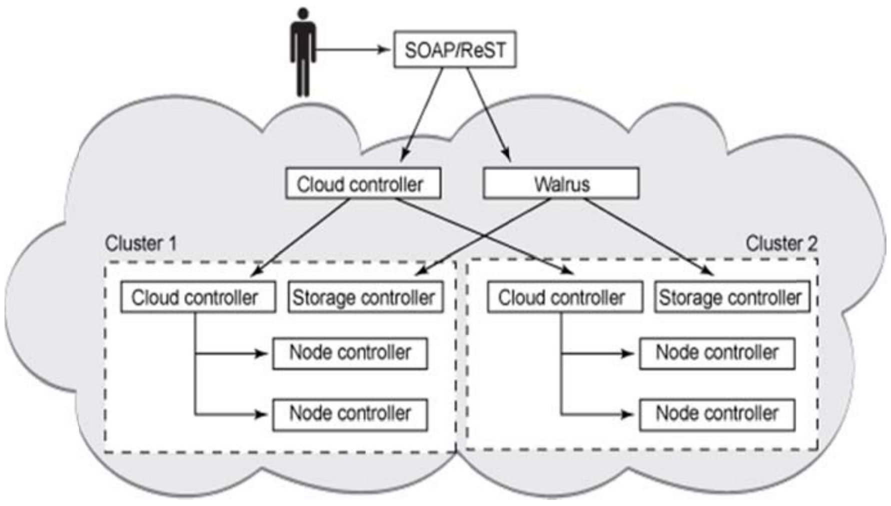


Figure 9. 5 Cluster Eucalyptus Installation Topologies[152153]

9.2.7 EUCALYPTUS Installation Readiness

Before installing EUCALYPTUS, it is critical to consider hardware requirements. For on testing purposes, we can run all content in a computer, but for actual deployment, many clusters are the better choice. The following index makes some suggested allocations of machine used to run CC, CLC, Walrus or SC.

Table 9. 1 Configuration for Machine used to run CC, CLC, Walrus or SC

Hardware	**Minimum**	**Recommended**
CPU	1 GHz	2 × 2 GHz
Memory	512 MB	2 GB
Disk	5400rpm IDE	7200rpm SATA
Disk space	40 GB	200 GB
Networking	100 Mbps	1000 Mbps

The above configuration is adequate for machines being used to run CC, CLC, Walrus or SC, but machines running as an NC need to be more powerful, as these machines will run every virtual example. They also need an adequate mount of disk space to store the image example. In the recommended configuration, the NC machine has a more powerful core with at least 4GB of memory and uses at least 7200rpm disk drives. The following are some Suggestions index.

Table 9. 2 Configuration for Machines used to Run NC

Hardware	MIN	Recommend
CPU	VT express	VT,64-Bit,Many nuclear
Memory	1 GB	4 GB
Disk	5400rpm IDE	7200rpm SATA or SCSI
Disk space	40 GB	100 GB
Networking	100 Mbps	1000 Mbps

9.2.8 Installation of EUCALYPTUS Technical Route

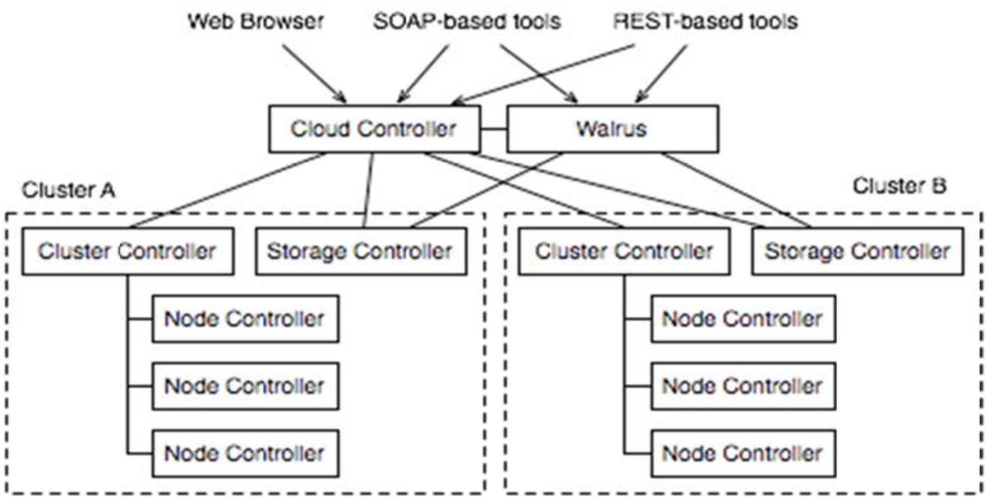


Figure 9. 6 Installation Structure Diagram

A Eucalyptus cloud setup consists of five types of components. The cloud controller (CLC) and "Walrus" are the top-level components, with one of each in a cloud installation. The cloud controller is a Java program that offers EC2-compatible SOAP and "Query" interfaces, as well as a web interface to the outside world. In addition to handling incoming requests, the cloud controller performs high-level resource scheduling and system accounting. Walrus is also written in Java and implements a bucket-based storage, which is available outside and inside a cloud viaS3-compatible SOAP and REST interfaces.

Top-level components can aggregate resources from multiple clusters (i.e., collections of nodes sharing a LAN segment, possibly residing behind a firewall). Each cluster needs a cluster controller (CC) for cluster-level scheduling and network control and a "storage controller"(SC) for EBS-style block-based storage. The two cluster-level components would typically be deployed on the head-node of a cluster (in fact, this is required if the cluster is behind a firewall). Finally, every node with a hypervisor will need a node controller (NC) for controlling the hypervisor. CC and NC are written in C and deployed as web services inside Apache with the SC are written in Java. Accordingly, communication among these components takes place over SOAP with WS-security.

Many instructions in this guide refer to a single-cluster installation, in which all components except NC are co-located on one machine, which we refer to as front-end. All other machines, running only NCs, will be referred to as nodes. In more advanced configurations, such as those with multiple CCs or with Walrus deployed separately, the front-end will refer to just the machine running the CLC.

9.2.8 Specific Methods of EUCALYPTUS Installation

We take the CPU supported by virtualization to install cloud platform based on EUCALYPTUS 1.6.2 (the latest version), Ubuntu 10.04 Server (the latest version). We used 3 computers as front and 8 computers as computing note that belonged to two clusters, using the 100 M switches to make an internal network. We used Ubuntu 10.04 server image phase, which includes SaaS application (with tomcat + MySQL + web service project application mirror) and uploading these mirrors to the cloud platform; make load equilibrium mirror that Ubuntu server10.04 mirror phase which has installed HAProxy (if we need many database instance, using MySQL cluster to achieve database load equilibrium). By accessing load equilibrium instance, dynamic scheduling and start with above mirror application allows our model of cloud computing to provide dynamic service.

By combining the characteristics of the Eucalyptus IaaS, IaaS allocate resources in the form of virtual machine. So we need to make the image with SaaS application (such as an Ubuntu 10.04 server or Windows 2003 server that install SaaS application (database and applications must be separated)). In addition, to achieve the distributed effect, we used a Load Balancing mirror phase (installed with HAProxy and finished configuration to an SaaS application) as a visiting entrance.. On the basis of the above measures, we wanted to make a cloud computing platform that is capable of automatic start up and closing. More poignantly, in our busy time we wanted the system to automatically increase new applications or database instances, while in our leisure time we wanted the system to close instances under the guarantee of service quality. By implementing this model, we would be able to save energy and release resources. Ultimately, the real goal was to realize a model of SaaS + PaaS + IaaS.

9.3 Cloud Computing Simulator: CloudSim in Use

CloudSim is a cloud computing simulation tool developed by the University of Melbourne, Australia. CloudSimis very powerful, as it not only supports multiple data centres but also multiple tasks handling. CloudSim provides a simulated platform useful in studying the distribution strategy of cloud resources..Using this simulation allows users to easily construct simulations of heterogeneous resources, users, and dispatcher and application programs, and so on. Meanwhile, it also can simulate interactive behaviour to complete the resource distribution process among many grid entities. Moreover, CloudSim also provides a great convenience for implementing our algorithms, since CloudSim supports the economic model based resource management mechanisms. To facilitate the work of developers, CloudSim also provides a rich API. A few further key features of CloudSim are also worth mentioning:

- Support for modelling and simulation of large-scale Cloud computing data centres;
- Support for modelling and simulation of virtualized server hosts, with customizable policies for provisioning host resources to virtual machines;
- Support for modelling and simulating energy-aware computational resources;
- Support for modelling and simulating federated clouds;
- Support for dynamic insertion of simulation elements, i.e., stop and resume simulation;
- Support for user-defined policies for the allocation of hosts to virtual machines and policies for the allocation of host resources to virtual machines[155]

9.4 The Structure of the CloudSim

CloudSim uses a hierarchical architecture to simulate the cloud computing, with tasks divided into the four levels where in each level focus on one particular problem. The structure can be described as follow in Fig. 9.7, which shows the layered implementation of the CloudSim software framework and architectural components:

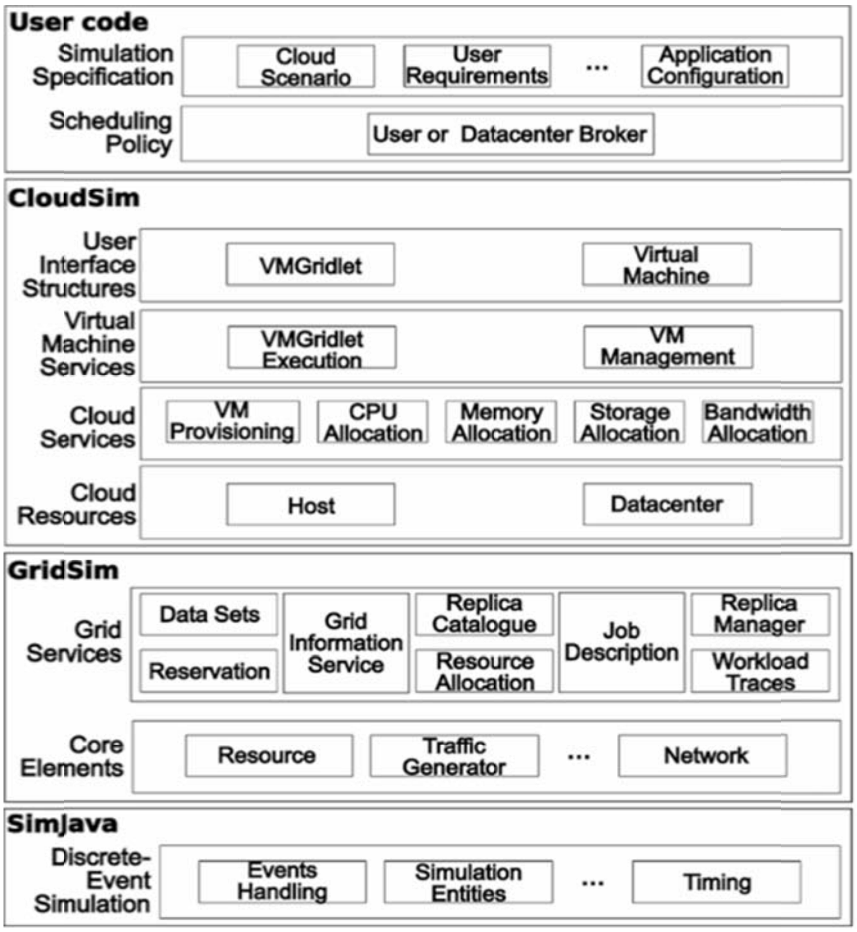


Figure 9. 7Structure of CloudSim[156]

At the lowest layer of CloudSim is the SimJava discrete event simulation engine that implements the core functionalities required for higher-level simulation frameworks such as queuing and processing of events, creation of system components (services, host, data centre, broker, virtual machines), communication between components, and management of the simulation clock. Next comes the libraries implementing the GridSim toolkit that supports high level software components for modelling multiple Grid infrastructures, including networks and associated traffic profiles, and fundamental Grid components such as the resources, data sets, workload traces, and information services.

The next layer in the hierarchy is a programmatic extension of the core functionalities exposed by the GridSim layer. CloudSim provides novel support for modelling and simulation of virtualized Cloud based data centre environments such as dedicated management interfaces for VMs, memory, storage, and bandwidth. CloudSim layer manages the instantiation and execution of core entities (VMs, hosts, data centres, application) during the simulation period. These layers can concurrently instantiate and transparently manage a large-scale Cloud infrastructure consisting of thousands of system components. This layer handles the fundamental issues—provisioning of hosts to VMs based on user requests, managing application execution, and dynamic monitoring. A Cloud provider, who wants to study the efficacy of different policies in allocating its host would need to implement their strategies at this layer by programmatically extending the core VM provisioning functionality. There is a clear distinction at this layer as to how a host is allocated to different competing VMs in the Cloud. A Cloud host can be concurrently shared among a number of VMs that execute applications based on user-defined QoS specifications.

The top-most layer in the simulation stack is the User Code that exposes configuration related functionalities for hosts (number of machines, their specification, and so on), applications (number of tasks and their requirements), VMs, number of users and their application types, as well as broker scheduling policies. A Cloud application developer can generate a mix of user request distributions, application configurations, and Cloud availability scenarios at this layer while also performing robust tests based on the custom Cloud configurations already supported within CloudSim.

9.5 Summary

IaaS provides customer service in all of its facilities, including processing, storage, network and other basic computing resources. Users can deploy and run any software, including the operating system and applications. The main reason we adopted EUCALYPTUS is that its design open programming interface (API) is compatible to the Amazon EC2 platform. This means that a company can use free software to build mutual compatibility test lab on a free operating system. At the same time, users can take EUCALYPTUS to engage in the development work before moving work to the actual cloud environment. At this moment then, EUCALYPTUS reduces the decision maker’s anxiety for cloud computing.

One fact that should be noted is that the EUCALYPTUS project is not completely open source. Some code is related to commercial version Eucalyptus Enterprise Edition (E3)'s characteristics, such as management, SAN integration, better backend database and VMware’s compatibility. OpenStack uses 100% open source products, but to date it has not provided the above characteristic functions that make EUCALYPTUS so attractive, but these function are currently included in its technology development plan (for more about the configuration process of EUCALYPTUS, refer to 8.2.4 and 8.2.5).

In order to verify the validity of the Cournot equilibrium pricing strategies as well as task scheduling algorithms, this chapter examined the experimental analysis of the proposed algorithm. In this chapter, we first analyzed the cloud computing simulation tool CloudSim in regards to its detailed modules and function. Second, this thesis introduced a detailed description of the resource scheduling process based on CloudSim. Third, we discussed the simulation of one of the aforementioned improved algorithms in CloudSim. Finally, in this chapter we also verified the improved resource scheduling algorithm for load balancing in fact improved both effectiveness and user satisfaction.

Chapter 10 Conclusions and Future Directions

10.1 Summary

The purpose of this research tries to deliver a new Cloud Bank model that can give a provoke discussion, collaboration and innovation that will push forward develop of public-based hybrid clouds. In its truest form, cloud computing organizes heterogeneous resources distributed geographically and belonging to different institutions into a virtual institution. Comprehensive sharing and collaboration in such a dispersed institution is difficult, largely due to the resources of the Cloud being owned by different organizations and resources scheduling strategies of every organization being different. To actually make such a system useable requires creating a system in which these resources can dynamically join and freely leave as needed, Cloud resource scheduling must shield the difficulties inherent in the system from the actual end-users.

Shielding users from the problems of using a distributed system is easier said than done. Resource exchange and service providers exist under a market environment, meaning that the lower level resources management and scheduling are influenced by forces of that same market economy. To that end, to build an effective model requires a greater understanding of those market forces. After studying both the economy theory these resources and providers exist within as well as the fundamental characteristic of cloud computing, this thesis were able to design a mechanism to manage and schedule the cloud resource based on the banking industry markets theory. This theory mainly focuses on how does the cloud computing follow the real bank to build the new cloud computing architectures.

This new framework Cloud Bank has deliver the whole bank as five levels and in each level also have some theories and algorithms to support them: Physical Resource Pool, SLA Pool, Risk Mitigation, Scheduling, and Pricing Policy. Throughout the course of composing this study, all the involved parts were developed, built, tested and refined. Likewise, these components were brought together into two open-source based cloud computing platform that allowed an ideal testing environment for some of our new related algorithms.

In summary, our work could be outlined as follows:

Pricing Policy

- For initialization phase: Cournot Equilibrium based pricing policy
- For stable phase: Game theory pricing policy
- For Updating: Deposit-loan ratio

Scheduling

- M × N Pareto Optimality based resource scheduling policy (PORS)
- Evolutionary game theory based load balance policy
- Resources select policy

Risk Mitigation

- Risk coping policy
- Risk predication policy

SLA pool

- It is not a real physical level, in the logical, all the resource and transaction have to base the discretion of the SAL. It will give all the necessary parameters and elements in the related entities.

Physical resource pool

- There is a real distributed resource that includes: CPU, memory, hard disk and network environment. They are Autonomy and join the hybrid Cloud Bank

10.2 Conclusions

As the first published framework for a Cloud Bank based on an economic theory based bank model to manage and scheduling cloud computing resources, this study represents a new step forward in utilizing the potentials of cloud computing. Compared with the traditional cloud resource management systems, this new framework will deliver a great value for all the related clouding computing users, one that is a dynamic composition system with simple processing modes capable of more easily sharing resources. This model offers is a novel chance to alter the ways in which we use and perceive of cloud computing, from academic research to commercial applications. A few key contributions are worth emphasizing:

1. Using a banking model for the cloud computing resource management offers a framework to develop a new ways to management cloud computing resource

exchange and services; such a deliver model has potential to change how we think of the cloud computing format.

2. The new resource pricing strategy and analysis of its constituent parties also offers exciting new possibilities. Both cloud resource providers and service consumers exist in a world built around marketing principles. To that end, our price adjustment algorithm of resource is divided into two categories. The Centralized synchronous algorithm uses the idea of total supply and demand balance, according to all the resources in the total supply and demand to simultaneously adjust the prices for all resources. This is the Cloud Bank-Centred adjustment algorithm, which is based on the optimal deposit-loan ratio theory to adjust the supply and demand balance. Likewise, the distributed price adjustment algorithm classifies resource first and then, based on the situation of each type of resource supply and demand, adjusts the price in order to achieve resource supply and demand balance. This is the Users-(resource provider and consumer) Centred adjustment algorithm. We first introduced the lift cycle concept in the cloud computing environment that divides the distributed price adjustment into two periods. The Cournot equilibrium is used to set the all price in initial period and then the Stackelberg model leads the price to an optimized state in the running time.
3. Though it is a well-established fact that the environment of cloud computing is dynamic and distributed, our analysis in this study has mapped out some of the uncertainties that may affect the Cloud Bank, which arise from resource providers being distributed physically across the globe with heterogeneous resources. According to some of certain features of the Cloud Bank. The strategy of risk mitigation in Cloud Bank is divided into two parts: prediction and management. Risks can be divided into the four categories: credit risk, liquidity risk, operational risk, and other risks. We mainly focus on the liquidity risk, operational risk and delivering a strategy to use mitigation those risks.
4. The Pareto optimality is called the “ideal kingdom” of fairness and efficiency. Pareto improvement can effectively achieve resource scheduling and allocation under the

condition wherein the system has to match consumers' requests (in here, namely, utility function) with available resources. Given that our Cloud Bank is established on the commercial bank model, transactions and distributing infrastructure resources are similar to the social resources assigned among social groups and individuals. Hence we can apply Pareto optimality theory to simulate the resource scheduling process in cloud computing. A formal description can be easily achieved by programming. We also extension the 2 × 2 Pareto optimality into the n × m and provided a secluding solution for the real cloud computing environment. Finally, there are some differences between cloud computing and economics practical. To guarantee the effectiveness of achieving resource scheduling in cloud computing (or QoS), people apply SLA. Various SLO in SLA describe rights and obligations of resource providers, resource consumers and even Cloud Bank itself.

10.3 Future Directions

This thesis equationted a distributed comical bank theory based cloud computing framework that supports the hybrid cloud computing environment in the real clouding environment. Likewise, this study demonstrated how the Cloud Bank works and how it can allow all the participants to achieve the maximum values from the cloud computing transaction.

10.3.1 Supporting accounting and Visualization

User resources and data information in the cloud and how to conduct audits, visualization, and accounting are among the most important directions for future studies. Cloud computing provides a virtual environment and users can access resources and services on demand. Users want to use the secure and reliable cloud services and would like to see transparent services and expected results. How to accurately measure the efficient use of resources as well as the result of customers' tasks is a very complex issue, and one in which we expect improvements on as the market for cloud computing evolves.

10.3.2 Supporting Complex service and task description

The ultimate goal of cloud computing is to provide services. But there are still some of problems such as the service description, i.e., how douser tasks match services, how do the services select the appropriate resources, and so on. This thesis is based on the SLA among depositors, lenders, and banks, but the simple description is still unable to meet the complex reality of the business environment at any given time. Because of the complexity of the resources itself in cloud computing, there are many performance indicators influencing each other when allocating resources. To that end, a more refined and tested scheduling policy and algorithm needs to be developed. We will continue in these improvements by further leveraging the economic models mentioned in this study.

10.3.3 Supporting Real Cloud Computing Environment Experiment Platform

Presently, only parts of the algorithm and strategy needed to make a fully functional and enterprise-level Cloud Bank have been tested in a simulation environment. In real hybrid cloud computing, many different participants with different resource will be making transactions. The next step logical is then is to build on the open source based EUCALYPTUS and OpenStack cloud computing platforms to build an improved experimental environment that can simulate actual user behaviour in the real environment, and continue to modify and improve the feasibility and effectiveness of the project.

10.3.4 Supporting a Variety of Risks in a Cloud Computing Environment

Risks can be divided into the four categories: credit risk, liquidity risk, operational risk, and other risks. Although we have credit risk and liquidity risk research, in a real trading environment, there are a lot of operational risk, which we simply had no way to test in the simulations we can. Only improvements to the designation of all risk types of analysis and strategy can make our Cloud Bank be more complete and ready to move forward.

Appendix A

Source Code of the Improved Algorithm

```
CloudSimulation.java
package org.cloudbus.cloudsim;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.core.SimEntity;
import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
import org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;
public class CloudSimulation {

    // The cloudlet list.
    private static List<Cloudlet> cloudletList;

    private static List<Cloudlet> temp_cloudletList;

    // The vmlist.
    private static List<Vm> vmlist;

    private static int vmNums;
    // The entities.
    private static List<SimEntity> entities;


     //Creates main() to run this example.
     // @param args

    public static void main(String[] args) {

        Log.printLine("The beginningofthesimulation program ！ ");

        try {
            //step1：Initialization
            int num_user = 1; // number of users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false;
            CloudSim.init(num_user, calendar, trace_flag);

// step2：Create a data centre, also on behalf of each resource provider
Map hostinfo = new HashMap();//Storage host description information, including memory
                                              size, disk space, etc.
int host_ram = 2048; // Host memory (MB)
long host_storage = 100000; //Host disk space (MB)
int mips=5000;
hostinfo.put("ram", host_ram);
hostinfo.put("storage", host_storage);
hostinfo.put("mips", mips);
Map costinfo = new HashMap();//Storage unitresourceprices, includingCPU,storage,
memory, etc.
double costPerCpu = 30.0; // A single CPUper hourto usetheprice
double costPerMem = 0.05; //Every Gigabytememoryper hourto usetheprice
double costPerStorage = 0.001; // The price of each gigabyte per hourdouble
costBw=0.005;//Bandwidth consumption
costinfo.put("cost", costPerCpu);
costinfo.put("costPerMem", costPerMem);
costinfo.put("costPerStorage", costPerStorage);
costinfo.put("bw", costBw);
int hostnum =1;// number of resource
int cores = 1;// number of CPU
Datacenter datacenter0 = createDatacenter("Datacenter_0", hostinfo,
                         costinfo, hostnum, cores);
Datacenter datacenter1 = createDatacenter("Datacenter_1", hostinfo,
                         costinfo, hostnum, cores);



// Step 3 create the agent responsible for the list of registered virtual machines, as well as the
task list
DatacenterBroker broker = createBroker();
int brokerId = broker.getId();

//Initialize task properties

long length = 40000;
long fileSize = 900;
long outputSize = 900;
int pesNumber=2;
TempCloudlet tempCloudlet=new TempCloudlet();
tempCloudlet.setLength(length);
tempCloudlet.setFileSize(fileSize);
tempCloudlet.setOutputSize(outputSize);
tempCloudlet.setPesNumber(pesNumber);
// Step 4: Creating virtual machine and the list of user's tasks
vmlist = createVM(brokerId,10); // Creating 20 virtual machines
cloudletList = createCloudlet(brokerId,9,tempCloudlet); // Creating 20 tasks
//Step 5, Submitted to the task and the current resources to CloudSim
broker.submitVmList(vmlist);
broker.submitCloudletList(cloudletList);

// Computing resources prices
//*********************************************
int providerNum = 10;
double K = 0;// The relationship between parametersoftheresourcesand price, expectto
                obtainthe best interests of
double bit = 1;
bit = 1.0 / providerNum;// The impactfactorofprice for the resource provideri

Account account = new Account();
account.accountCostPrice(hostinfo, costinfo, cores);
K = (1 + 0.2) * account.getCostPrice();//Expect to get thecurrentcost ofan
                                          additional20%interest
          System.out
.println("************************************************");
      System.out.println("The price of the resource cost "+account.getCostPrice());
       System.out.println("Resources toprovidethose whoexpectthe best
                             interests of:" + K);
   account.accoutResourceNum(providerNum, K, bit);
       System.out.println("Resources to providethe amount of resources:" +
                             account.getResourceNum());
      account.accoutResourcePrice(K);
      System.out.println("Price oftheactualuse of resources:" + account.getPrice());
      double value = 0.0;
      double unitDebt=0.0;
```

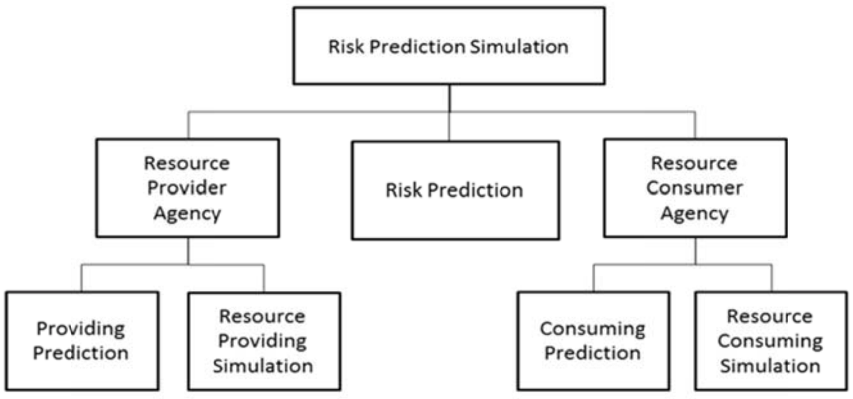
```
            double estimatedFinishTime=0.0;//Estimated task completion time
            double estimatedDebt=0.0;//Estimated thatthe task is completedtheprice
            double minEstimatedDebt=0.0;//Estimated a minimumprice
            //*********************************************


// Seek current task corresponding execution time on all the virtual machines
Account tempaccount=new Account();

String indent = "";
System.out.println("/*******The list of task completion time ******/");
System.out.print("Cloudlet ID"+indent);
      for(int j=0;j<vmlist.size();j++)
       {
            System.out.print("VM"+j+indent);
      }
System.out.println(indent);
temp_cloudletList=cloudletList;

int i=-1;

//**************************************************************
double [] vmImplementTime=new double[vmlist.size()];//Storage resources to the task
execution time
double spaceTime=0;//Task arrival interval
int j=0;
for(Cloudlet cloudlet:cloudletList)
{
      System.out.print(cloudlet.getCloudletId()+indent+indent+indent);
      or(Vm vm:vmlist)
        {
             tempaccount.setEstimatedFinishTime(cloudlet.getCloudletLength(),
         vm.getMips(), vm.getPesNumber(), cloudlet.getPesNumber());
      value=tempaccount.accountVmCost(vm,
        cloudlet,costinfo,hostinfo,cloudlet.getPesNumber());
                          account.accountResourceDebt(value);
                          unitDebt=account.getUserDebt();
                          if(spaceTime<vmImplementTime[j]){

      estimatedFinishTime=tempaccount.getEstimatedFinishTime()+vmImplementTime[j]-spa
ceTime;
      }else{
      estimatedFinishTime=tempaccount.getEstimatedFinishTime();
                          }

      estimatedDebt=unitDebt*estimatedFinishTime/3600;//Prices withintheunithours
                          estimatedDebt=Math.round(estimatedDebt*100)/100.0;
                          if(i==-1){
                               minEstimatedDebt=estimatedDebt;
                               i++;
                          }
                          if(minEstimatedDebt>estimatedDebt){
                               minEstimatedDebt=estimatedDebt;
                               i=vm.getId();
                          }
//                   System.out.print("m*:"+unitDebt);
//                   System.out.print("n*:"+estimatedFinishTime+":");
                          System.out.print(estimatedDebt+indent);
                          j++;
                     }
                     broker.bindCloudletToVm(cloudlet.getCloudletId(),i);
                System.out.println(indent);
                spaceTime+=100;// Arrival time of the next task
                vmImplementTime[i]=estimatedFinishTime;
                j=0;
                    i=-1;
                }
                System.out
                .println("**************************************************");

// step 6, Start to perform the task
CloudSim.startSimulation();
CloudSim.stopSimulation();

// step 7, output the result
List<Cloudlet> newList = broker.getCloudletReceivedList();
printCloudletList(newList);

// Print the debt of each user to each datacenter
datacenter0.printDebts();
datacenter1.printDebts();
Log.printLine("The endofthesimulation experiment!");

            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }


        // Creates the datacenter.
        // @param name
        // @return the datacenter

        private static Datacenter createDatacenter(String name, Map hostinfo,
                Map costinfo, int hostnum, int cores) {

            // Here are the steps needed to create a PowerDatacenter:
            // We need to create a list to store our machine
            List<Host> hostList = new ArrayList<Host>();
            int hostId;
            int ram;
            long storage;
            int bw = 10000;
            for (int i = 0; i < hostnum; i++) {// Register multiple hosts
                // 2. A Machine contains one or more PEs or CPUs/Cores.
                List<Pe> peList = new ArrayList<Pe>();
                int mips = (Integer)hostinfo.get("mips");
                for (int j = 0; j < cores; j++) {// Register multiple cores
                    // 3. Create PEs and add these into a list.
                    peList.add(new Pe(j, new PeProvisionerSimple(mips)));
                }
                // 4. Create Host with its id and list of PEs and add them to the
                // list
                // of machines
                hostId = i;// Host numbers
                ram = (Integer) hostinfo.get("ram");// Host memory（MB）
                storage = (Long) hostinfo.get("storage");// Host disk space（MB）

                hostList.add(new Host(hostId, new RamProvisionerSimple(ram),
                        new BwProvisionerSimple(bw), storage, peList,
                        new VmSchedulerTimeShared(peList))); // This is our machine

            }
            // 5. Create a DatacenterCharacteristics object that stores the
            // properties of a data centre: architecture, OS, list of
            // Machines, allocation policy: time- or space-shared, time zone
            // and its price (G$/Pe time unit).
            String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String vmm = "Xen";
            double time_zone = 100.0; // time zone this resource located
            double cost = (Double) costinfo.get("cost"); // the cost of using
                                                                    // processing in this
                                                                    // resource
            double costPerMem = (Double) costinfo.get("costPerMem");
  // the cost of using memory in this resource
            double costPerStorage = (Double) costinfo.get("costPerStorage");
// the cost of using storage in this resource
            double costPerBw =(Double)costinfo.get("bw"); // the cost of using bw in this
resource
            LinkedList<Storage> storageList = new LinkedList<Storage>();
  // we are not adding SAN devices by now

            DatacenterCharacteristics characteristics = new DatacenterCharacteristics(
                        arch, os, vmm, hostList, time_zone, cost, costPerMem,
                        costPerStorage, costPerBw);

            // 6. Finally, we need to create a PowerDatacenter object.
            Datacenter datacenter = null;
            try {
                  datacenter = new Datacenter(name, characteristics,
                              new VmAllocationPolicySimple(hostList), storageList, 0);
            } catch (Exception e) {
                  e.printStackTrace();
            }

            return datacenter;
      }

      /**
        * Creates the broker.
        *
        * @return the datacenter broker
        */
      private static DatacenterBroker createBroker() {
            DatacenterBroker broker = null;
            try {
                  broker = new DatacenterBroker("Broker");
            } catch (Exception e) {
                  e.printStackTrace();
                  return null;
            }
            return broker;
      }

      //
        // Prints the Cloudlet objects.
        // @param list of Cloudlets

      private static void printCloudletList(List<Cloudlet> list) {
            int size = list.size();
            Cloudlet cloudlet;

            String indent = "";
            Log.printLine();
            Log.printLine("========== OUTPUT ==========");
            Log.printLine("Cloudlet ID" + indent + "STATUS" + indent
                        + "Data centre ID" + indent + "VM ID" + indent + "Time"
```

```
                + indent + "Start Time" + indent + "Finish Time");

        DecimalFormat dft = new DecimalFormat("###.##");
        for (int i = 0; i < size; i++) {
            cloudlet = list.get(i);
            Log.print(indent + cloudlet.getCloudletId() + indent + indent);

            if (cloudlet.getCloudletStatus() == Cloudlet.SUCCESS) {
                Log.print("SUCCESS");

                Log.printLine(indent + indent + cloudlet.getResourceId()
                        + indent + indent + indent + cloudlet.getVmId()
                        + indent + indent
                        + dft.format(cloudlet.getActualCPUTime()) + indent
                        + indent + dft.format(cloudlet.getExecStartTime())
                        + indent + indent
                        + dft.format(cloudlet.getFinishTime()));
            }
        }
    }

    public static double getVmDebt(Datacenter datacenter) {
        Set<Integer> keys = datacenter.getDebts().keySet();
        Iterator<Integer> iter = keys.iterator();
        DecimalFormat df = new DecimalFormat("#.##");
        double value = 0.00;
        while (iter.hasNext()) {
            int key = iter.next();
            value = datacenter.getDebts().get(key);

        }
        return value;
    }

    private static List<Cloudlet> createCloudlet(int userId, int cloudlets,TempCloudlet
tempcloudlet){
        // Creates a container to store Cloudlets
        LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();

        //cloudlet parameters
        long length = tempcloudlet.getLength();
        long fileSize = tempcloudlet.getFileSize();
        long outputSize = tempcloudlet.getOutputSize();
        int pesNumber = tempcloudlet.getPesNumber();
        UtilizationModel utilizationModel = new UtilizationModelFull();

        Cloudlet[] cloudlet = new Cloudlet[cloudlets];

        for(int i=0;i<cloudlets;i++){
            length=(long)(Math.random()*50000);
            pesNumber=(int)(Math.random()*10);
            fileSize=(long)(Math.random()*1000);
            outputSize=(long)(Math.random()*1000);
            cloudlet[i] = new Cloudlet(i, length, pesNumber, fileSize, outputSize,
utilizationModel, utilizationModel, utilizationModel);
            // setting the owner of these Cloudlets
            cloudlet[i].setUserId(userId);
            list.add(cloudlet[i]);
        }

        return list;
    }
    private static List<Vm> createVM(int userId, int vms) {
        //Creates a container to store VMs. This list is passed to the broker later
        LinkedList<Vm> list = new LinkedList<Vm>();

        //VM Parameters
        long size = 10000; //image size (MB)

        int ram = 512; //vm memory (MB)
        int mips = 1000;
        long bw = 1000;
        int pesNumber = 1; //number of cpus
        String vmm = "Xen"; //VMM name

        //create VMs
        Vm[] vm = new Vm[vms];

        for(int i=0;i<vms;i++){
            ram=(int) (Math.random()*1024);
            mips=(int)(Math.random()*1000);
            vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, new
CloudletSchedulerTimeShared());
            //for creating a VM with a space shared scheduling policy for cloudlets:
            //vm[i] = Vm(i, userId, mips, pesNumber, ram, bw, size, priority, vmm, new
CloudletSchedulerSpaceShared());

            list.add(vm[i]);
        }

        return list;
    }

}
Account.java
package org.cloudbus.cloudsim;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

/* Calculating the number of computing resources, and resource prices * */
public class Account {

    int resourceNum=0;//Amount of resources
    public int getResourceNum() {
        return resourceNum;
    }
    public int getProviderNum() {
        return providerNum;
    }
    public double getCostPrice() {
        return costPrice;
    }
    public int getRam() {
        return ram;
    }
    public long getStorage() {
        return storage;
    }
    public int getCores() {
        return cores;
    }
    public double getCostPerMem() {
        return costPerMem;
    }
    public double getCostPerStorage() {
        return costPerStorage;
    }
    public double getPrice() {
        return price;
    }
    public double getUserDebt() {
        return userDebt;
    }
    int providerNum=1;//number of the resource providers
    double costPrice=0.0;
    int ram=0;
    long storage=0;
    int cores=1;
    double costPerMem=0.000;
    double costPerStorage=0.000;
    double price=0.000;// the price of resource
    double userDebt=0.000;//the price of users need to pay
double estimatedFinishTime=0.0;// Estimated completion time oftasksin a virtual machine

    /* Calculate a single resource costs */
    public void accountCostPrice(Map hostinfo,Map costinfo,int cores){
        //Map cost=new HashMap();
        ram=(Integer)hostinfo.get("ram");//Host memory（MB）
        storage=(Long)hostinfo.get("storage");//Host disk space（MB）
        this.cores=cores;
        costPerMem =(Double)costinfo.get("costPerMem") ;
        costPerStorage =(Double)costinfo.get("costPerStorage");
        costPrice+=ram*costPerMem;
        costPrice+=storage*costPerStorage;
        //cost.put("costPrice", costPrice);
        //return cost;

    }
    //According to the Cournot equilibrium, calculate the amount of resources provided by
each resource        * /
        public void accoutResourceNum(int providerNum,double k,double bit){
        double temp=0.0;
        temp=(k-costPrice)/(1+bit);
        resourceNum=(int)Math.floor(temp);
    }

    // Calculate the balanced resources prices */
    public void accoutResourcePrice(double k){

        price =k-resourceNum;
    }

  //Calculate the user is currently usingtheresource pricespayableintheresources
    public void accountResourceDebt(double vmcosts){
        userDebt=vmcosts*price/costPrice;
        userDebt=Math.round(userDebt*100)/100.0;
    }

    //Calculation of the current task in the current virtual machine on the task execution time

    public double getEstimatedFinishTime() {
        return estimatedFinishTime;
    }

    public void setEstimatedFinishTime(long cloudletLength,double capacity,int
vmPesNumber,int cloudletPesNumber){
        estimatedFinishTime=(cloudletLength*cloudletPesNumber)/(capacity*vmPesNumber);

        }
        // Calculation of the cost of the use of virtual machines within the current time
        public double accountVmCost(Vm vm,Cloudlet cloudlet,Map costinfo,Map hostinfo,int
cloudletPesNumber){
                double vmcost=0.0;
                vmcost+=vm.getRam()*(Double) costinfo.get("costPerMem");
                vmcost+=vm.getSize()*(Double) costinfo.get("costPerStorage");
                vmcost+=cloudlet.getCloudletFileSize()*(Double)costinfo.get("bw");
                vmcost+=cloudlet.getCloudletOutputSize()*(Double)costinfo.get("bw");

        vmcost+=100*vm.getMips()*(Double)costinfo.get("cost")/(Integer)hostinfo.get("mips");
                return vmcost;
        }

        // Get a list of tasks sorted
        public List<Cloudlet> getSortList(List<Cloudlet> cloudletList){
                long tempLengths[]=new long[cloudletList.size()];
                List<Cloudlet> list=new ArrayList<Cloudlet>();
                long tempLength=0;
                for(int i=0;i<cloudletList.size();i++){
                for(Cloudlet cloudlet:cloudletList){
                        tempLength=cloudlet.getCloudletLength();
                        if(i==0){
                                tempLengths[i]=tempLength;
                                list.add(cloudlet);
                        }
                        if(tempLengths[i]>tempLength){
                                tempLengths[i]=tempLength;
                                list.add(cloudlet);
                        }
                }
                }
                return list;
        }
}
TempCloudlet.java
        package org.cloudbus.cloudsim;

        public class TempCloudlet {

                long length = 40000;
                long fileSize = 900;
                long outputSize = 900;
                int pesNumber=1;
                public long getLength() {
                        return length;
                }
                public void setLength(long length) {
                        this.length = length;
                }
                public long getFileSize() {
                        return fileSize;
                }
                public void setFileSize(long fileSize) {
                        this.fileSize = fileSize;
                }
                public long getOutputSize() {
                        return outputSize;
                }
                public void setOutputSize(long outputSize) {
```

```
            this.outputSize = outputSize;
        }
        public int getPesNumber() {
            return pesNumber;
        }
        public void setPesNumber(int pesNumber) {
            this.pesNumber = pesNumber;
        }
}
```

Appendix B

Key Source Code of Risk Prediction Strategy Algorithm



Simulation block diagram

```
publicclass Provider_Simulation {
    public ArrayList<Resource_Info> ResourceAddHistory(ArrayList<Resource_Info>
resHistory){
        // Randomly generate the type of resource, the range is 1 to 4, corresponding to the
digital resource type relationship is:
//1:CPU
        //2:Memory
        //3:HardDisk
        //4:Network
        long resType = Math.round(Math.random()*3+1);
        // Randomly generate the type of resource, the range is 100 to 1000
        long resNum = Math.round(Math.random()*900+100);
        // Produce a resource trading, resource type and quantity corresponding
        Resource_Info resInfo = new Resource_Info();
        resInfo.setResource_Type((int)resType);
        resInfo.setResource_Num((int)resNum);
        resHistory.add(resInfo);
        return resHistory;
    }
    // A determination of the correction factor in the exponential smoothing model
    publicfloat Parameter_Determine(ArrayList<Resource_Info> ali){
        float a = 0.1f;
        float []b = null;
        float init,record = 0,sum=0,min=Float.MAX_VALUE;
        int i,swi=1,j = 0;
        init=(float)ali.get(0).getResource_Num();
        b[0]=init;
        while(swi==1&&j<ali.size()){
            sum=0;
            for(i=1;i<=ali.size();i++)
            {
                    // The variance number s corresponding to the value of the calculated 
correction parameter
                    b[i]=a*((float)ali.get(i).getResource_Num())+(1-a)*b[i-1];
                }
                for(i=0;i<ali.size();i++)
                {

    sum+=(b[i]-(float)ali.get(i).getResource_Num())*(b[i]-(float)ali.get(i).getResource_Nu
m());
                }
                // The variance number s corresponding to the value of the calculated correction 
parameter
                if(sum<=min){
                    min=sum;
                    record=a;
                }
                // The change of the correction parameters a gradient of 0.1
                a+=0.1f;
                j++;
            }
            a=record;
            return a;
    }

    // Forecasting process
    publicfloat[]ResourcePrediction(){
        int i=1;
        // Correction parameters for each category of resources set exponential smoothing 
models
        float Para_CPU,Para_Memory,Para_HardDisk,Para_Bandwidth,tem;
        // The final result of each category of resources based on exponential smoothing 
models
        float Ad_CPU,Ad_Memory,Ad_HardDisk,Ad_Bandwidth;
        float []Results = null;
        // Initialize the various types of resources
        ArrayList<Resource_Info> aliHis = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliCPU = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliMemory = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliHardDisk = new ArrayList<Resource_Info>();
        ArrayList<Resource_Info> aliBandwidth = new ArrayList<Resource_Info>();
        while(i<=100){
            ResourceAddHistory(aliHis);
            i++;
        }
        // Resource classified
        for(Resource_Info resInfo:aliHis){
            if(resInfo.getResource_Type()= =1){
                aliCPU.add(resInfo);
            }
            if(resInfo.getResource_Type()= =2){
                aliMemory.add(resInfo);
            }
            if(resInfo.getResource_Type()= =3){
                aliHardDisk.add(resInfo);
            }
            if(resInfo.getResource_Type()= =4){
                aliBandwidth.add(resInfo);
            }
        }
        // Determination of the correction parameters of various types of resources
        Para_CPU = Parameter_Determine(aliCPU);
        Para_Memory = Parameter_Determine(aliMemory);
        Para_HardDisk = Parameter_Determine(aliHardDisk);
        Para_Bandwidth = Parameter_Determine(aliBandwidth);
        // Forecast results to determine all kinds of resources
        // The predicted results of the CPU
        tem=(float)aliCPU.get(0).getResource_Num();
        for(i=1;i<aliCPU.size();i++){
            tem=(float)aliCPU.get(i).getResource_Num()*Para_CPU+(1-Para_CPU)*tem;
        }
        Ad_CPU=tem;
        // the predicted results of the memory
        tem=(float)aliMemory.get(0).getResource_Num();
        for(i=1;i<aliMemory.size();i++){

    tem=(float)aliMemory.get(i).getResource_Num()*Para_Memory+(1-Para_Memory)*te
m;
        }
        Ad_Memory=tem;
        // the predicted results of the HardDisk
        tem=(float)aliHardDisk.get(0).getResource_Num();
        for(i=1;i<aliHardDisk.size();i++){
    tem=(float)aliHardDisk.get(i).getResource_Num()*Para_HardDisk+(1-Para_HardDisk)*
tem;
        }
        Ad_HardDisk=tem;
        // the predicted results of the Bandwidth
        tem=(float)aliBandwidth.get(0).getResource_Num();
        for(i=1;i<aliBandwidth.size();i++){

    tem=(float)aliBandwidth.get(i).getResource_Num()*Para_Bandwidth+(1-Para_Bandwid
th)*tem;
        }
        Ad_Bandwidth=tem;
        Results[0]=Ad_CPU;
        Results[1]=Ad_Memory;
        Results[2]=Ad_HardDisk;
        Results[3]=Ad_Bandwidth;
        return Results;
    }
```

```
}
```

Appendix C

Selected Relevant Publications

1. **Hao Li**, Gehao Lu, Joan Lu ,Shaowen Yao . "*A Grid Resources Allocation Mechanism Based on Banking Industry Markets Theory*". The first DBAT2007 of CSS, computer science,2007 :198-106.
2. **Hao Li**, Gehao Lu, Joan Lu, Shaowen Yao. *"mobile embed based LNIC real estate presell system"*.The first DBAT2007 of CSS, computer science, 2007
3. **Hao Li**, Gehao Lu, Joan Lu , Shaowen Yao . "*PDA based web service in real estate presell system*". Annual conference of network committee of CCS, computer science, 2007
4. **Hao Li**, Gehao Lu ,Joan Lu , Shaowen Yao. *"Semantic Enhanced Self-Configuring Grid Framework"* (**EI Index)**.The 2006 International Conference on Internet Computing (ICOMP'06: July 14-17, 2006)
5. **Hao Li**, Gehao Lu, Joan Lu , Shaowen Yao. "*An Investigation in Grid Resources Allocation Mechanism Based on Banking Industry Market Theory*". Journal of Computer Science (China)vol(34),10A,2007:263-266.
6. **Hao Li**, Gehao Lu,li Li.*"A research on Multi-Agent Systems based on trust negotiation mechanism"*. Annual conference of network committee of CCS, computer science, 2007:138-143.
7. **Hao Li**, Joan Lu, Ning Gong, Dexian Li. *"Investigation of XQuery mechanism in Water Supply Information System"*. Proceedings of the 13th International Conference on Automation and Computing, 15 September 2007, Staffordshire University.
8. **Hao Li**, Joan Lu, Dexian Li, Gang Xue.*"Mappings between BPMN and XPDL Based on XML-binding"*.Proceedings of the 13th International Conference on Automation and Computing, 15 September 2007, Staffordshire University.
9. **Hao Li**, Joan Lu, Xuejie Zhang, Shaowen Yao.*"A Banking Based Grid Recourse Allocation Scheduling"*(**EI Index)** The 3rd International Conference on Grid and Pervasive Computing, China.Published by IEEE Computer Society, May, 2008:239-244.
10. **Hao Li**, Joan Lu, and Gehao Lu. "*The Study and Actualization of Key Technologies in WebGIS Based on XML*". The 2008 International Conference on Internet Computing (ICOMP'08: July 14-17, 2008)
11. **Hao Li**, Kebing Tang, Rongfei Zhao. *"Design and implementation of analysis based on the Eclipse platform plug-in development"*. Annual conference of network committee of CCS, computer science, 2008.

12.**Hao Li**,Yu Liu, Joan Lu, Shaowen Yao. "*OLAP Model System for Reporting Use"*.Proceedings of the 14th International Conference on Automation & Computing, Brunel University, West London, UK, 6 September 2008.

13.**Hao Li**, Joan Lu, Binjing Cai, Shaowen Yao. "*Study on SOA-Orient WebGIS Framework"*. Proceedings of the 14th International Conference on Automation & Computing, Brunel University, West London, UK, 6 September 2008.

14.**Hao Li**, Joan Lu, Guo Tang, Bingjing Cai. "*Grid Bank Model Based QoS Grid Resource Quantization"*. The 2009 International Conference on Internet Computing. July. 2009:56-64.

15.**Hao Li**, Joan Lu, Guo Tang, Bingjing Cai. "*A Study on Grid Resource Bidding Method Based on Historical Utilization Rate"*.The 2009 International Conference on Internet Computing .July, 2009:220-228.

16. **Hao Li**, Guo Tang, Joan Lu, and Saowen Yao.*"Based on banking model QoS cloud resources quantify"*. Journal of Computer and Digital EngineeringVol.37(9), 2009:29-36

17.**Hao Li**, Guo Tang, Joan Lu, and Saowen Yao."*The QoS Resource Quantification Based on the Grid Banking Mode"*. (**EI Index)** The First International Conference on 'Networked Digital Technologies'(NDT2009), Ostrava, Czech Republic, during July 28-31.Published by IEEE Computer Society. P.548-553

18.**Hao Li**, Guo Tang, Wei Guo, Changyan Sun, Shaowen Yao.*"Price-oriented Trading Optimization for Grid Resource"* (**EI Index)**.The First International Conference on Cloud Computing, Beijing, during Dec. 1-4. 2009. Published by Springer LNCS 5931:P650-655.

19. **Hao Li**, Kai Song BoWang. *"Grid bank-based optimal deposit and loan resources management strategy"*. Journal of Computer Science. Vol.37(8), 2010.8:182-184

20.**Hao Li**, Hong Li, Gehao Lu, and Joan Lu. "*A Logistics Service Platform Based on SOA"*. The 2010 International Conference on Internet Computing (ICOMP'10: July 12-15, 2010)

21**.Hao Li**, Ran Chen, Bo Wang, Bing Kong.*"The Research of Grid Resource Scheduling Mechanism Based on Pareto Optimality"*. The 2010 International Conference on Internet Computing (ICOMP'10: July 12-15, 2010)

22.**Hao Li**, Bo Wang, Bing Kong.*"The Pricing Scheme of the Grid Resource Based on Cournot equilibrium"*. (**EI Index)** 2010 Second World Congress on Software Engineering (WCSE 2010)

23. **Hao Li**, Rong liu, Guo Tang. *"The Pricing Resources under the Cloud Banking Model"* (**EI Index)**. The 3nd International Conference on Computer and Network Technology (ICCNT 2011), Taiyuan Feb. 26-28. 2011.V3 122-126 Published by IEEE
24. **Hao Li**, Jianhui Liu,Guo Tang. *"A Pricing Algorithm for Cloud Computing Resources"* (**EI Index)** 2011 International Conference on Network Computing and Information Security (NCIS'11) Guilin May 14-15. Published by IEEE CS. ISBN 978-0-7695-4355-0 P69-74 P69-74
25. **Hao Li**, Guo Tang.*"Pareto-based Optimal Scheduling on Cloud Resource"* (**EI Index)** Springers CCIS Proceddings of ICHCC-ICTMF 2011, Singapore. The First International Conference on Cloud Computing, Beijing, during May 25-26 2011. Published by Springer LNCS
26. **Hao Li**, Huixi Li.*" A Research of Resource Scheduling Strategy For Cloud Computing Based on Pareto Optimality M×N Production Model"*(**EI Index)**. International Conference on Management and Service Science (MASS 2011), during August25-26 2011. Published by IEEE CS
27. **Hao Li**, Fengchuan Zhu. *"A Stackelberg Leadership Model Based Cloud Resource Pricing Strategy"*(**EI Index)**. 2011 International Conference on Intelligent Computing and Integrated Systems during Oct. 24-26 2011.Guilin China Published by IEEE CSP638-642.
28. **Hao Li**, Miao Xin. *"An Approach for Cloud Resource Risk Prediction"*(**EI Index)** 2012 International Workshop on Information and Electronics Engineering (IWIEE) China Published Elsevier. Available online at www.sciencedircet.com
29. **Hao Li**, Maojun Su. *"A Research of the Perdition of the Liquidity Risk in Cloud Computing"*(**EI Index)**.The 2nd International Conference on Computer and Management (CAMAN 2012)2012 China Published IEEE CS.

Bibliography

[1]Giorgos Cheliotisy, C.K., Rajkumar Buyya, "*Grid Economics: 10 Lessons from Finance*", in Joint Technical Report. 2003.in Joint Technical Report. 2003.

[2]Hao Li, Y.Z., Joan Lu,Xuejie Zhang,Shaowen Yao.*"A Banking Based Grid Recourse Allocation Scheduling"* GPC Workshops. 2008.

[3]Fawaz, W.; Daheb, B.; Audouin, O.; Du-Pond, M.; Pujolle, G.; *"Service level agreement and provisioning in optical networks"*.Communications Magazine, IEEEJ an 2004. Volume: 42 Issue:1 page:36-43.

[4]Ilias tsagklis *"Advantages and Disadvantages of Cloud Computing – Cloud computing pros and cons"* http://www.javacodegeeks.com/2013/04/advantages-and-disadvantages-of-cloud-computing-cloud-computing-pros-and-cons.html

[5]www.eucalyptus.com. Retrieved 2011-10-22

[6]www.ubuntu.com. Retrieved 2011-10-22

[7]Eric Rasmusen,*"Games and information"*, Blackwell Publishers, 1997.

[8]William Novshek, *"Cournot Equilibrium with Free Entry "*.Review of Economic Studies(1980)XL VII,473-486

[9]Rabha Amin,Isabel Grilo, *"Stackelberg versus Cournot Equilibrium"*.Games and Economic Behavior. Volume 26, Issue 1, January 1999, Pages 1–21

[10]Harold M. Hochman and James D. RodgersThe American Economic Review Vol. 59, No. 4, Part 1 (Sep., 1969), pp. 542-557

[11]Fudenberg, D.; Tirole, J. (1983)."1.2.4 Multiple Nash Equilibria, Focal Points, and Pareto Optimality".Game Theory. MIT Press. p. 18. ISBN 0-262-06141-4.

[12]Carroll, Sue; Daughtrey, Taz. *"Fundamental Concepts for the Software Quality Engineer"*. American Society for Quality(July 4, 2007).pp. 282 et seq. ISBN 978-0-87389-720-4.

[13]Blaise Barney. Lawrence Livermore National Laboratory, *"Introduction to Parallel Computing"* , http://computing.llnl.gov/tutorials/parallel_comp/.

[14]Parallel Computing, *"Introduction to Parallel Computing"*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©2002 ISBN:0201648652

[16]Andrews, Gregory R. (2000), *"Foundations of Multithreaded, Parallel, and Distributed Programming"*, Addison–Wesley, ISBN 0-201-35752-6 .

[17]Arora, Sanjeev; Barak, Boaz, " *Computational Complexity – A Modern Approach"*, Cambridge, 2009, ISBN 978-0-521-42426-4 .

[18]Prof. Y.K.Shah & Supriya SHAH *"The Dynamic IT Architecture of Grid Computing in Modern Applications"* Int. J Comp Sci. Emerging Tech.Vol-3 No 4 August, 2012

[19]I-WAY.*"Information-Wide-Area-Year"* http://www.nitrd.gov/pubs/bluebooks/1997/i-w.html. Retrieved 2010-08-22

[20]Michael Litzkow, Miron Livny, and Matt Mutka, *"Condor - A Hunter of Idle Workstations",* Proceedings of the 8th International Conference of Distributed Computing Systems, pages 104-111, June, 1988

[21]Ian Foster, *"Globus: a Metacomputing Infrastructure Toolkit"*. International Journal of High Performance Computing Applications June 1997 vol. 11 no. 2 115-128.

[22]Antony Rowstron. *"Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility"*.SOSP '01 Proceedings of the eighteenth ACM symposium on Operating systems principles.ACM New York, NY, USA ©2001. Volume 35 Issue 5, Dec. 2001

[23]Li-Guohui; Luo-Tiejian; Song-Jinliang; Xu-Yanxiang .*"A Security Model for Online Accessing to Shared Devices"*.Networking, Sensing and Control, 2006.ICNSC '06.Page on 143-149.

[24]"*tweettele immersion"*. http://www.webopedia.com/TERM/T/tele-immersion.html Retrieved 2011-07-22.

[25]Eric Knorr, Galen Gruman, *"What cloud computing really means"*. Published on InfoWorld. Created 2008-04-07 02:00AM

[27]*"Gartner's 2011 Hype Cycle Special Report Evaluates the Maturity of 1,900 Technologies"* .http://www.gartner.com/it/page.jsp?id=1763814. Retrieved 2011-07-22

[28]*"Gartner's 2011 Hype Cycle Special Report Evaluates the Maturity of 1,900 Technologies"* .http://www.gartner.com/it/page.jsp?id=1763814. Retrieved 2011-07-22

[29]Dan Farber, *"on-demand computing: what are the odds?"*. ZANet, Nov 2003, retrieved Oct 2010

[30]Mauri. *"AppFirst Launches First SaaS-Based Application Performance Monitoring Solution For Entire Application"* http://www.networkcomputing.com/data-center/229501827. Retrieved 2011-04-15

[31]Ye Wei ,el, *"the revolution of the Software in the Internet age—SaaS Architecture design"* Electronic Industry Press ,China. ISBN：9787121077364.P137

[32]Mariana Carroll, Paula Kotzé, Alta van der Merwe (2012). *"Securing Virtual and Cloud Environments"*. In I. Ivanov et al. Cloud Computing and Services Science, Service Science: Research and Innovations in the Service Economy. Springer Science+Business Media.doi:10.1007/978-1-4614-2326-3.

[33]Danielson, Krissi (2008-03-26). *"Distinguishing Cloud Computing from Utility Computing"*. Ebizq.net. Retrieved 2010-08-22

[34]*"Gartner Says Cloud Computing Will Be As Influential As E-business"*. Retrieved 2010-08-22

[35]NIST.gov – *"Computer Security Division - Computer Security Resource Center"*. Retrieved 2010-08-22.

[36]http://www.allbusiness.com/technology/407675-1.html. Retrieved 2010-04-28

[37]https://docs.google.com/. Retrieved 2010-04-25

[38]http://zh.wikipedia.org/zh/Microsoft_Office_Live. Retrieved 2010-04-25

[39]http://www.auciti.com/home.html. Retrieved 2010-04-30
[40]http://code.google.com/intl/zh-CN/appengine/. Retrieved 2010-05-01
[41]http://www.microsoft.com/windowsazure/. Retrieved 2010-05-01
[42]*"Joyent Buys Reasonably Smart"* http://www.bloomberg.com/apps/news?pid=newsarchive&sid=aSTqj1.EpS2o. Retrieved 2010-05-22
[43]Dyan Machan. *"The New Information Goldmine"* http://online.wsj.com/article/SB125071202052143965.html. Retrieved 2010-05-22
[44]"Data as a Service: Are We in the Clouds?". Journal of Map & Geography Libraries6 (1): 76–78. January 2010. Retrieved 2010-06-09.
[45]*"Data as a Service: Are We in the Clouds?"*.Journal of Map & Geography Libraries6 (1): 76–78. January 2010. Retrieved 2010-06-09.
[46]Lizhe Wang; Jie Tao; Kunze, M.; Castellanos, A.C.; Kramer, D.; Karl, W.; *"Scientific Cloud Computing: Early Definition and Experience"*.High PerformanceComputing and Communications, 2008. HPCC '08. On page(s): 825 - 830
[47]Dyche, Jill. *"Data-as-a-service, explained and defined"*. SearchDataManagement.com. Retrieved October 24, 2010.
[48]Zoltan Nagy, United Nations. *"Statistical Data as a Service and Internet Mashups"* . Retrieved 2010-06-09.
[49]Cagle, Kurt. *"Why Data as a Service Will Reshape EAI"*. DevX.com. Retrieved 2010-06-13
[50]Hong-Linh Truong; Dustdar S.*"On analyzing and specifying concerns for data as a service"*. Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific on page:87-94
[51]Sushil Bhardwaj, Leena Jain, Sandeep Jain, *"A Study of Infrastructure as a service"* Sushil Bhardwaj and Etal, IJEIT 2010, 2(1), 60-63
[52]http://cloudcomputing-vision.com/cloud-computing/iaas/. Retrieved2010-06-02.
[53]http://aws.amazon.com/. Retrieved 2010-06-15.
[54]http://www.ibm.com/cloud-computing/us/en/. Retrieved 2010-06-15.
[55]http://www.rackspace.com/. Retrieved 2010-06-15.
[56]Nuno Santos ,Krishna P. Gummadi, Rodrigo Rodrigues. *"Towards trusted cloud computing"*. HotCloud'09 Proceedings of the 2009 conference on Hot topics in cloud computing.
[58]Rajkumar Buyya, James Broberg and Andrzej Goscinski *"Cloud Computing: Principles and Paradigms"*, Copyright r 2011 John Wiley & Sons, Inc.
[60]Ian Forster, Carl Kesselman, *"The Grid 2: Blueprint for a New Computing Infrastructure"*, Morgan Kaufmann; 2nd edition (Novermber 18,2003):319-351
[61]Menasce, D.A.; Casalicchio, E.; *"QoS in grid computing"*.Internet Computing, IEEE. July-Aug. 2004. 8 Issue:4. On page(s): 85 – 87. ISSN: 1089-7801
[62]Rajkumar Buyya, James Broberg and Andrzej Goscinski *"Cloud Computing: Principles*

and Paradigms", Copyright r 2011 John Wiley & Sons, Inc.
[63]http://wiki.filezilla-project.org/Network_Configuration
[64]Rajkumar Buyya, James Broberg and Andrzej Goscinski *"Cloud Computing: Principles and Paradigms"*, Copyright r 2011 John Wiley & Sons, Inc.
[66]IBM cloud computing white paper[EB/OL]. Retrieved 2010-06-15
[68]http://research.microsoft.com/en-us/projects/dryad/. Retrieved 2011-02-15
[70]Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: *"Distributed data-parallel programs from sequential building blocks"*. In: Proc. Of the 2nd European Conf. on Computer Systems(EuroSys). 2007.59-72.
[71]http://zh.wikipedia.org/zh/MapReduce. Retrieved 2011-03-04.
[72]Peter Buneman, "*A query language and optimization techniques for unstructured data"*.SIGMOD '96 Proceedings of the 1996 ACM SIGMOD international conference on Management of data. ACM New York, NY, USA ©1996 ISBN:0-89791-794-4
[73]Jeffrey Dean and Sanjay Ghemawat jeff. *"MapReduce: Simplied Data Processing on Large Clusters"* @google.com, sanjay@google.com Google, Inc. Retrieved 2011-03-04.
[74]Jeffrey Dean and Sanjay Ghemawat jeff. *"MapReduce: Simplied Data Processing on Large Clusters"* @google.com, sanjay@google.com Google, Inc. Retrieved 2011-03-04.
[76]Zach Hill, Jie Li, Ming Mao, Arkaitz Ruiz-Alvarez, Marty Humphrey. *"Early observations on the performance of Windows Azure"*. HPDC '10 Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing. ISBN: 978-1-60558-942-8.
[77]http://research.microsoft.com/en-us/projects/dryad/. Retrieved 2011-03-04.
[78] Brivand BS *"A Walk Through on Windows Azure"* Retrieved 2012-05-24.
[79]Introducing_Windows_Azure_v1-Chappell.pdf. Retrieved 2011-03-04.
[80]Nurmi, D.; Wolski, R.; Grzegorczyk, C.; Obertelli, G.; Soman, S.; Youseff, L.; Zagorodnov, D.;*"The Eucalyptus Open-Source Cloud-Computing System"*.Cluster Computing and the Grid, 2009. CCGRID '09.
[81]The AWS Cloud Architecture Best Practices White Paper http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf
[82]Jeffrey Dean，Sanjay Ghemawat. "*MapReduce:Simplified Data Processing on Large Clusters"*[C]．OSDI,2004
[84]Shvachko, K.; Hairong Kuang; Radia, S.; Chansler,*"The Hadoop Distributed File System"*.Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium onIssue Date: 3-7 May 2010.Page 1-10.
[85]Shafer, J.; Rixner, S.; Cox, A.L.; *"The Hadoop distributed filesystem: Balancing portability and performance"*.Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium onIssue Date: 28-30 March 2010.Page 122-133.
[86]hadoop-presentation/Hadoop-AWS.pdf. Retrieved 2011-03-04
[88]Konstantin Shvachko, Hairong Kuang, Sanjay Radia, Robert Chansler "*The Hadoop Distributed File System"*, California USA {Shv, Hairong, SRadia, Chansler}

@Yahoo-Inc.com

[89]www.eucalyptus.com. Retrieved 2011-05-04

[90]http://www.marketwire.com. Retrieved 2011-05-04

[91]http://www.eucalyptus.com/pdf/whitepapers/Eucalyptus_Overview.pdf. Retrieved 2011-05-04

[92]http://www.eucalyptus.com/pdf/whitepapers/Eucalyptus_Overview.pdf

[93]Yohan Wadia (2012). *"The Eucalyptus Open-Source Private Cloud"*. Cloud book. Retrieved June 1, 2013.

[94] *"Intel® Cloud Builder Guide to Cloud Design and Deployment on Intel® Platforms".* Retrieved 2011-06-14

[95]http://www.virtualizationtechnology.com. Retrieved 2011-05-04

[96]K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, *"A Resource Management Architecture for Metacomputing Systems"*, Proceedings of the 4thInternational Workshop on Job Scheduling Strategies for Parallel Processing, March 30, 1998,Orlando, Florida, USA, Springer Verlag Press, Germany, 1998

[97]Andrew Grimshaw, Hiro Kishimoto. *"Open Grid Services Architecture WG (OGSA-WG)"*. http://forge.gridforum.org/sf/projects/ogsa-wg. Retrieved 2011-06-14

[98]*"Defining an open framework for modeling and accessing stateful resources using Webservices"*.http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf. Retrieved 2011-06-14.

[99]Xue zhiwei,Li wei, *"Research on the Vega Grid Architecture . Journal of computer research and development"*, Volume 39, Number 8, Aug. 2008

[100]Michael R. Ault, Mike Ault, Madhu Tumma, and Ranko Mosic.*"Oracle 10g Grid & Real Application Clusters"*.(2004)Rampant Tech Press. p. 24. ISBN0974435546.

[101]D.H.J. Epema, M. Livny, R. van Dantzig, X. Evers, J. Pruyne *"A worldwide flock of Condors: Load sharing among workstation clusters"* .Volume 12, Issue 1, May 1996, Pages 53–65 ELSEVIER

[102]R. Wolski, J. Plank, J. Brevik, and T. Bryan, *"Analyzing Market-based Resource Allocation Strategies for the Computational Grid"*, International Journal of High-performance Computing Applications, Volume 15, Number 3, Sage Publications, USA, Fall 2001.

[103]http://www.zetagrid.net/. Retrieved 2011-06-14

[104]Chris Kenyon and Giorgos Cheliotis, *"Architecture requirements for commercializing grid resource"*, Proceeding of the 11th IEEE international symposium on high performance distributed computing, 2002 page 123-133.

[105]JunYao， Manfu Ma, *"Resource binding in grid computing"*. Journal of Computer Engineering and Design(China), 2007. 28(22): p. 5562-5564.

[106]JunYang, *"A research of grid environment based on the pricing model the distribution of storage"*.Journal of Harbin University of Commerce (Natural Science)volume 23 issue 5 2007. 23(5): p. 564-567.

[107]Hao Li, Y.Z., Joan Lu,Xuejie Zhang,Shaowen Yao. *"A Banking Based Grid Recourse Allocation Scheduling"*. The 3rd International Conference on Grid and Pervasive Computing- Workshops. 2008. Kunming, China: IEEE CS.

[108]Peijie Huang, H.P., Piyuan Lin, Xuezhen Li, *"Macroeconomics based Grid resource allocation"*. Future Generation Computer Systems, 2008. 24(7): p. 694-700.

[109]Tang Xiaoyong, Li kengli, *"Based on trusted grid economics scheduling algorithm"* Journal of Application Research of Computers (China), 2008. 25(8): p. 2357-2361.

[110]Rajkumar Buyya, David Abramson, Jonathan Giddy and Heinz Stockinger *"Economic models for resource management and scheduling in Grid computing"* CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2002; 14:1507–1542 (DOI: 10.1002/cpe.690)

[111]Xing Tang, Maojun Li, *"Agent-based electricity marketbiddinggridmodel"*. Journal of Computing Technology and Automation (China), 2008. 27(1): p. 16-20.

[112]Pu Wang, Ling Peng, *"A new economic grid computing task scheduling and control model"*. Journal of Computer Science (China), 2008. 35(3): p. 106-108.

[113] *"Grid Economic Services Architecture"* (GESA).http://www.ggf.org/Meetings/ggf7/drafts/CompEconArch1.doc, 2006-05-02. Retrieved 2011-08-14

[114]Bingquan He, Microeconomics. 2005, Chongqing University Press.

[115]Yuanpeng Zhang, Microeconomics tutorial. 2005, Beijing,China Development Press

[116]Grid Economic Services Architecture (GESA).http://www.ggf.org/Meetings/ggf7/drafts/CompEconArch1.doc, 2006-05-02. Retrieved 2011-09-18

[117]Samuelson, Paul A.; Nordhaus, William D. *Microeconomics* (17th ed.) (2001).. McGraw-Hill. p. 110. ISBN 0071180664.

[118]Bingquan He, Microeconomics. 2005, Chongqing University Press.

[119]Ting Li, Xiaolong Li, *"A research on the cloud resource management"*. Journal of Computer and telecom (China). Issue 1, 2010 page 35-45.

[120]P. Mell and T. Grance, "*The NIST Definition of Cloud Computing, National Institute of Standards and Technology, Information Technology Laboratory"*, Technical Report Version 15, 2009.

[122]http://www.investopedia.com/terms/l/loan-to-deposit-ratio.asp

[123]Xiangfei Li, *"Optimal deposit-loan ratio of commercial bank: experience model and empirical analysis"*. Journal of Banking &Finance , 2006 ,27, 2323-2345;

[124]P.J.Green， Silverman. "*Nonparametric regression and generalized linear models"*， Chapman＆Hall， 1994

[126]Douglas W. Diamond and Raghuram G. Rajan, *"Liquidity Risk, Liquidity Creation and Financial Fragility: A Theory of Banking"*, National Bureau of Economic Research Working Paper No.7430, Cambridge MA 02138, December 1999.

[127]Rajkumar Buyya, *"Economic-based distributed resource management and scheduling*

for Grid computing", Thesis.2002, Monash University.

[128]Mark Tranmer Mark Elliot "Multiple Linear Regression", the lecture notes of CCSR Form http://www.ccsr.ac.uk/publications/teaching/mlr.pdf

[129]Baki Billaha, Maxwell L. Kingb, Ralph D. Snyderb, and Anne B. Koehlerc, *"Exponential smoothing model selection for forecasting"*, International Journal of Forecasting, vol22, pp.239-247, April-June 2006.

[130]David W. Hosmer and Stanley Lemeshow, *"Applied logistic regression"*, 2rd ed.,vol 1.Canada:John Wiley and Sons, 2000, pp.1-31.

[132]Rajkumar Buyya, *"Economic-based distributed resource management and scheduling for Grid computing"*, in Thesis. 2002, Monash University.

[133]Katherine Dusak, *"Future Trading and Investor Returns: An Investigation of Commodity Market Risk Premiums "*.Journal of Political Economy. Vol.81. No.6.Nov-Dec.1973

[134]Eric Rasmusen, *"Games and information"*, Blackwell Publishers, 1997.

[135]Narahari, Y.; Garg, Dinesh; Narayanam, Ramasuri; Prakash, Hastagiri (2009), *"Game Theoretic Problems in Network Economics and Mechanism Design Solutions"*, Springer, p. 21, ISBN 978-1-84800-937-0

[136]Hao Li,Guo Tang,Joan Lu ,Shaowen Yao, *"The QoS Resource Quantification Based on the Grid Banking Model"*, Proceedings of the First International Conference on 'Networked Digital Technologies' (NDT 2009) Ostrava, The Czech Republic, July 29 – 31, 2009.ISBN:978-1-4244-4615-5.pp.548-553 (published by IEEE CS)

[137]Bo Wang,Yuanyuan Miao,Bing Kong, Hao Li.*"The researarch on cloud resource pricing strategies based on Cournot equilibrium,"* CSIE2011, Changchun, China. 17 – 19 June.(EI Compendex, ISTP, and IEEE Xplore indexed)

[138]Hao Li, Fengchuan Zhu. *"A Stackelberg Leadership Model Based Cloud Resource Pricing Strategy"*. 2011 International Conference on Intelligent Computing and Integrated Systems during Oct. 24-26 2011.Guilin China Published by IEEE CS P 638-642

[139]Ludwig H, Keller A, Dan A, King R and Franck R(2003). *"Web service level Agreement (WSLA) language speciation"*. IBM Corporation.

[140]Peter Mell,Timothy Grance, *"Recommendations of the National Institute of Standards and Technology"*. Special Publication 800-145 (Draft).

[141]Hao Li, Guo T, Joan L, Shaowen Y(2009). *"The QoS Resource Quantification Based on the Grid Banking Model"*, 2009.ISBN:978-1-4244-4615-5.pp.548-553

[142]Hao Li, Ran Chen, Bo Wang, Bing Kong.*"The Research of Grid Resource Scheduling Mechanism Based on Pareto Optimality"*. The 2010 International Conference on Internet Computing (ICOMP'10: July 12-15, 2010)

[144]Hao Li, Guo T, Joan L, Shaowen Y(2009). *"The QoS Resource Quantification Based on the Grid Banking Model"*, 2009.ISBN:978-1-4244-4615-5.pp.548-553

[146]A. MC. Michael, M. D. Whinston, and J. R. Green, *"Microeconomic Theory",* Oxford

University Press, 1995, pp.9.

[147]Rajkumar Buyya, J. B., and Andrzej Goscinski, *"Computing Principles and Paradigms"*, A JOHN WILEY & SONS,INC., 2010, pp.

[148]P Patal, A Ranabahu, *"Service Level Agreement in Cloud Computing"*, ACM International Conference, 2009.

[149]Sushi Bhardwaj, Leena Jain, Sandeep Jain, *"Cloud Computing: "A Study of Infrastructure as a Service (IaaS)".* International Journal of Engineering and Information Technology Vol 2 , No. 1Copyright© 2010 waves publishers

[150]Kai Hwang, Geoffrey C.Fox Jack J. Dongarra. *"Distributed and Cloud Computing form Parallel Processing to the Internet of Things"* Publish by Elsevier Inc. Page 277.ISBN-13: 978-0-12-385880-1

[151]http://open.eucalyptus.com/wiki/EucalyptusInstall_v1.6. Retrieved 2011-09-18

[153]http://open.eucalyptus.com/wiki/EucalyptusInstall_v1.6. Retrieved 2011-09-18

[155]Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, Cesar A. F. De Rose, and Rajkumar Buyya, *"CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms"*, Software: Practice and Experience (SPE), Volume 41, Number 1, Pages: 23-50, ISSN: 0038-0644, Wiley Press, New York, USA, January, 2011. (Preferred reference for CloudSim)

[156]Anton Beloglazov, and Rajkumar Buyya, *"Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers, Concurrency and Computation: Practice and Experience"*, ISSN: 1532-0626, Wiley Press, New York, USA, 2011, DOI: 10.1002/cpe.1867