Resource management in the Cloud: An End-to-end Approach

By:

Kun Ma (Makuning@126.com)

A thesis submitted in partial fulfillment of the requirments

for the degree of

Doctor of Philosophy

Department of Computer Science,

University of the Western Cape

Supervisors:

Prof. Antoine Bagula

May 2020

Acknowledgements

At this moment, a myriad of gratitude fill my heart. I would like to take this opportunity to give my sincere thanks for those who offered the generous help during my four years of PhD studies.

First of all, I would like to thank my supervisor, Professor Antoine Bagula, for his inspirational guidance and instructive guidance that have helped me greatly in the past year. His professionalism inspires me all the time, which guides me how to be a real researcher. He not only gave me a lot of help on searching the interesting problem and papers writing, but, more importantly, showed me how to conduct the scientific research with critical thinking as well. Thanks for your encouragement and all discussions we made. You are my life-time comrade and I wish you all the best for the future.

Secondly, I would like to thank Dr. Ajayi and Dr. Nyirenda for their valuable guidance. They provided the real important supports for the papers we worked together, and also lots of valuable comments and suggestions to improve my work.

Thirdly, I would like to especially thank our ISAT team, we have the regular meeting per week, we do the presentation per week, we discuss to each per week, and we share our idea, our resource together. We are united, ISAT team is our strong backing.

Last but not the least, I dedicate this thesis to my parents and my girlfriend, you are the fulfilment of my lonely moment, the driving force of my advance, and the harbour of my soul. My gratitude to your love is beyond my words, I will always love you.

Publications

1. Ma K, Bagula A, Mauwa H, et al. Modelling Cloud Federation: A Fair Profit Distribution Strategy Using the Shapley Value[C] 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud). IEEE, 2018: 393-398.

2. Ma K, Bagula A, Nyirenda C, et al. An IoT-Based Fog Computing Model[J]. Sensors, 2019, 19(12): 2783.

3. Xu, Lingyu, et al. "Design of a Credible Blockchain-Based E-Health Records (CB-EHRS) Platform." 2019 ITU Kaleidoscope: ICT for Health: Networks, Standards and Innovation (ITU K). IEEE, 2019.

4. Ajayi, Olasupo O., Antoine B. Bagula, and Kun Ma. "Fourth Industrial Revolution for Development: The Relevance of Cloud Federation in Healthcare Support." *IEEE Access* 7 (2019): 185322-185337.

5. Ma K, Bagula A, Ajayi O, Nyirenda C. Aiming at QoS: A Modified DE Algorithm for Task Allocation in Cloud Computing. InICC 2020-2020 IEEE International Conference on Communications (ICC) 2020 Jun 7 (pp. 1-7). IEEE.

Abstract

Cloud Computing enables users achieve ubiquitous, on-demand, and convenient access to a variety of shared computing resources, such as servers, networks, storage, applications, and more. As a business model, Cloud Computing has been openly welcomed by users and has become one of the research hotspots in the field of information and communication technology. This is because it provides users with on-demand customization and pay-per-use resource acquisition methods. The heterogeneity of compute nodes in Cloud data centre, the dynamic and massiveness of user task requests, and the increasing size of Cloud data centres, have brought about the challenge of task scheduling and virtual machine management. This challenge has received wide attention from industry and academia in recent times. Despite the numerous research works, key issues still remain unresolved, prominent among which are: i.) Cloud resource providers tend to share their resources among multiple concurrent services owned by different customers. This practice requires sophisticated resource management mechanisms that dynamically manage the provider's resources in the most cost-effective manner, yet delivering on expected quality of service (QoS) levels agreed with the customers. ii.) Cloud users have varied QoS requirements, hence, comprehensively considering all QoS targets during task scheduling in a challenge and an active research area. iii.) Existing research works on virtual machine management usually selects the virtual machine (VM) to be migrated according to its resources occupation and the number of migrations. However, an important factor of virtual machine migration overhead is often overlooked. This results in proposed models being able to minimize overall energy consumption in Cloud data centres but at the detriment of high migration overhead. iv.) Due to the nature of Cloud Computing itself, there are some inevitable limitation in the communication between the Cloud layer and the underlying layer (Internet of Thing layer). These include latency, bandwidth and power consumption issues. Aiming at the presented challenges, this thesis conducts an in-depth research on resource allocation in different layers of Cloud Computing, and proposes corresponding models and algorithms to address the

challenges.

The main research work and contributions of the thesis are reflected in the following aspects:

- To address the issue of resource sharing, Cloud federation has been proposed by this thesis. An analysis of the necessary conditions for the composition and maintenance of the Cloud federation is done, after which a fairness profit sharing strategy is proposed. This strategy is able to measure the contribution of each participant by using the economic concepts of Shapley Value; which aims at fairly distribute the profit between members of a group/team in accordance to their contributions. A comprehensive fair profit distribution method, not only ensures the federation stability, but also attracts more Cloud service providers to join the federation. Finally, the fairness of our proposed strategy has been proved by the experimental results.
- For the second challenge, a multi-QoS target constraint-based task collaborative scheduling strategy is proposed. In Cloud Computing environments resources are dynamic and varied, while user preferences are diverse with multiple metrics. The users' satisfaction on the QoS largely determines the performance of Cloud task scheduling strategy. A task scheduling strategy proposed in this thesis targets the QoS constraint requirements of different user tasks by combining the modified Differential Evolution algorithm with Shapley Value. Compared with the traditional DE and Cloud-sim task banding policy, the proposed method can effectively reduce the deadline of the user task scheduling, improve the comprehensive QoS performance of the system, while satisfying the multi-QoS target constraint requirements of the user task.
- In addressing the third challenge，a VM migration-aware model was developed and was achieved in two phases: i.) proposal of a virtual machine consolidation with migration overhead awareness; ii.) proposal of a virtual machine consolidation with both cooperative and competitive Cloud federations. This

algorithm reduces the consolidation overhead in the virtual machine migration process by selecting the virtual machine with the smallest overhead factor to migrate. Finally, experimental results show that the proposed algorithm can reduce the virtual machine migration overhead, and how it works with different types of Cloud federations.

- For the fourth challenge, this thesis proposed an IoT-based Fog Computing model, which took into consideration the delay, distance and energy consumption between the fog and terminal layers. A modified routing protocol for the IoT layer was designed for data collection and muling. Furthermore, a modified genetic algorithm (GA) was proposed to solve the problem of resource allocation on fog nodes. Finally, experiments were used to prove that the modified routing protocol model proposed performed better in terms of robustness and sensor energy control. Similarly, at the fog layer, the modified GA showed better performance when compared to the classic MaxMin algorithm and the fog oriented MaxMin algorithm.

Key words: Cloud Computing, Evolution algorithm, Fog Computing, Internet of things (IoT), LIBP, Quality of service (QoS), Profit distribution, Resource allocation, Virtual machine migration, Shapley Value.

Directory

Chapter 1: Introduction

1.1 Research Background and Significance

Cloud Computing is a new business computing model and a product of the ongoing information technology revolution. The resource types of Cloud Computing systems are usually heterogeneous and dynamic in nature. Additionally, there are large numbers of Cloud users, with diverse quality of service (QoS) requirements. This series of factors makes the task scheduling and resource allocation problem in the Cloud Computing environment particularly complex. However, due to the nature of Cloud Computing itself, certain challenges are inevitable, such as those relating to delay (as a result of physical distance between users and the Cloud data centres) and energy consumption when communicating with the Internet of Things. In view of the above description, this thesis will proceed from several aspects such as modelling Cloud federation by a fair profit distribution strategy, the QoS target constraints of Cloud Computing task scheduling, the migration necessity-based virtual machine migration technology and an IoT-based Fog Computing model. This section introduces the research background and significance of this thesis.

1.1.1 The concept of "Cloud Computing"

The birth of Cloud Computing is the product of the information technology revolution. With the rapid development of information technology and the increasing network bandwidth, the demand for computing resource and storage is on the rise. The traditional computing model could no longer meet people's urgent needs of high-performance computing or massive data storage space. In this context, grid computing technology was developed and it matured rapidly. By integrating a large number of idle computing resources via the Internet, grid computing can effectively deal with all kinds of complex scientific computing problems, and it can also well meet the needs of a large number of institutions or individuals for high-performance computing applications. However, because grid computing technology is more focused on solving large-scale scientific computing problems, it is not well applied in the field of commercial computing. Therefore, Cloud Computing technology has emerged as an extension of grid computing technology in commercial applications. The concept of "Cloud Computing" was first proposed by the Google Corporation, United States in 2006; as a new business computing service model. It is the result of mixed evolution of multiple technologies. It inherits the technical foundations of grid computing and combines it with utility computing and software as a service (SaaS) [1]. Grid computing laid the technical foundation for resource sharing and resource integration for the development of Cloud Computing technology [2]. The utility computing uses services as a quantifiable commodity, upon which the business service model for Cloud Computing was developed; while SaaS provides a concrete and feasible commercial billing plan for Cloud Computing technology. It is precisely because of the above-mentioned existing technological foundations and the vigorous promotion of many large companies that Cloud Computing has developed rapidly upon its introduction. It has achieved lots of excellent application cases in many commercial fields and generated tremendous influence in its short existence [3]. There is no universally accepted statement about the definition of "Cloud Computing”. Wikipedia considers Cloud Computing to be a business computing model. Specifically, it is a dynamic, scalable, virtualized resource provided over the Internet and in the form of a service. In [4], Cloud Computing is considered as a virtual resource pool containing a large amount of available resources. These resources may be hardware resources, software resources, network resources or development platforms; the entire virtual resource pool is provided by Cloud service providers. The whole resource pool is based on the principle of pay-as-you-go, and its maintenance and management are performed according to a Service Level Agreement (SLA) [5]. This resource pool allow dynamic configuration for the purpose of optimization. Foster, *et al.* [6] believe that Cloud Computing technology is a large-scale distributed commercial computing model driven mainly by economic factors. It provides users with a virtual resource pool made up of a large number of computers based on virtualization technology, so that users can obtain all kinds of computing resources on demand.

1.1.2 Features of Cloud Computing

In summary, Cloud Computing technology has the following characteristics:

1. The huge server size. At present, the Cloud Computing platforms of IT giants such as Google, Amazon, IBM, Microsoft and Yahoo, usually have hundreds of thousands or even millions of servers, while private Cloud projects of IT companies generally have hundreds or thousands of servers. The large scale of servers in Cloud Computing systems can provide system users with unprecedented computing power and storage space.

2. Virtualization. Through virtualization technology, resources distributed in different geographic locations are integrated into logically unified resource pools. Users can access the services provided by Cloud Computing systems at any time and any place via Internet. The resources requested by users come from the logical Cloud. They do not have to care about the specific locations where these resources are deployed. Virtualization is both the foundation and an important feature of Cloud Computing [7].

3. Reliability and scalability. In order to maintain the cost advantage, a large number of cheap equipment is often used to deploy the server node of the Cloud Computing system, resulting in frequent failures in the Cloud Computing system and severe single point failure. For this reason, Cloud Computing systems usually ensure the reliability of Cloud Computing systems by introducing various fault-tolerant mechanisms, such as replica strategies and node isomorphism interchange technologies [8]. On the other hand, the resources of Cloud Computing systems are dynamically scalable and their size can be dynamically adjusted based on users'

demands. This scalability avails users the ability to purchase Cloud Computing resources and services of any size according to their needs.

4. Cost Efficient/Afforabilty. The economy of scale used in Cloud Computing, lowers the prices of computing resource. This offers significant cost advantage to users versus purchasing and setting up private servers [9]. In addition, integrating all types of IT resources are integrated for unified deployment through mature virtualization technologies, can realize automatic control and optimization management of system resource usage, thereby providing users with a transparent service, and Cloud services can be billed flexibly like water and electricity [10].

5. Versatility. Cloud Computing can provide a wide range of service content, and its service is not limited to specific types of applications. In addition, Cloud Computing can not only support multiple types of applications, but also can execute application computing, data storage, video playing and other types of applications at the same time.

6. User-centric. Cloud Computing provides a huge resource pool, which enable users only need to install a Cloud Computing client on the local terminal to obtain services from the Cloud Computing system. In this process, users do not need to change their original work habits or work environment, such as operating systems, programming languages, and so on.

The purpose of Cloud Computing is to share resources and work collaboratively. However, due to the large scale of Cloud Computing servers, resources are heterogeneous and dynamic. On one hand, they provide services to a wide range of users, hence require scheduling of resources. On the other hand, the types of tasks are varied, and the QoS target constraint requirements are different. This series of factors makes the task scheduling and resource allocation problem in the Cloud Computing environment very complicated. The task scheduling problem in the Cloud Computing environment has certain similarities with the task scheduling in the traditional distributed environment, but there are also great differences. Firstly, unlike in traditional distributed computing environment, the resources in the Cloud system are dynamical, and resources can be added or removed at any time. The Cloud Computing task scheduling strategy must be able to monitor resources changes in real time. Secondly, the types of resources in the Cloud Computing environment are heterogeneous and oblivious to the users. They are shielded through mature virtualization technologies, integrated into a unified logical resource pool and jointly provided as external services. This is in contrast to traditional distributed environments where computing resources are often homogeneous. In addition, unlike the traditional distributed environment, the Cloud Computing task scheduling policy is generally not limited to a specific application. It can support multiple types of applications and can run multiple applications at the same time. Finally, the task scheduling task in the traditional distributed environment is relatively simple. It only pays attention to the overall performance index of the traditional distributed system environment, such as task completion time and system throughput. In Cloud Computing environment, the task scheduling strategy also seeks to improve the service revenue of Cloud service providers as well as provide cater for the sharing of profit between multiple collaborative providers. It must do these while also satisfy the requirements of a large number of users for different resource types and QoS target constraints for different scheduling tasks. Furthermore, it must effectively manage the allocation of virtual resource (vms) in a manner that conserves energy. The inclusion of a Fog Computing layer between Cloud data centre and IoT layer (to decrease delay, control bandwidth and save energy), further complicates the task of the Cloud scheduler.

1.1.3 Research significance of Cloud Computing resource scheduling

Cloud Computing uses virtualization technology to consolidate a large number of IT resources, such as servers, computing clusters, network facilities, and software systems, distributed in different regional locations into a logically unified virtual resource pool. It aims at providing a large number of users with all kinds of safe and reliable, low-cost, simple delivery, highly scalable computing or storage service on the way of "pay-as-you-go" [11]. Cloud Computing users can purchase Cloud Computing resources or services of any size as needed, without bothering about the actual physical location of such resources. To the Cloud service users, the Cloud Computing resources are perceived as unlimited [12]. The goal of Cloud Computing is to realize resource sharing and collaborative work. However, due to the large scale of Cloud Computing servers, the resources are heterogeneous and dynamic, and on the other hand, it provides services to the general public, which has a wide user base, diverse tasks requests types. These series of factors makes the task scheduling problem in the Cloud Computing environment very complicated. The task scheduling problem in the Cloud Computing environment has certain similarities with the traditional task scheduling, and there are also great differences. First of all, the resources in the Cloud system are dynamic, and new resources will be added to the Cloud Computing system at any time. At the same time, existing resources may exit the system at any time. The Cloud Computing task scheduling strategy may cost the critical waste of computing resources, serious affect Cloud Service provider's revenue [13].

In summary, the Cloud Computing system has a large server scale, diverse resources, a wide user base, different types of application tasks, and different requirements for service quality objectives. The Cloud Computing system must handle a large number of user tasks and massive data at all times [14]. In this context, balancing these multiple requirements and objectives, has become a research hotspot and technical difficulty in the field of Cloud Computing. [15]. Therefore, in the Cloud-based business-based business computing model, the in-depth study of its resource scheduling strategy not only has high theoretical value, but also has good practical significance.

1.2 Cloud Computing resource allocation and the challenges

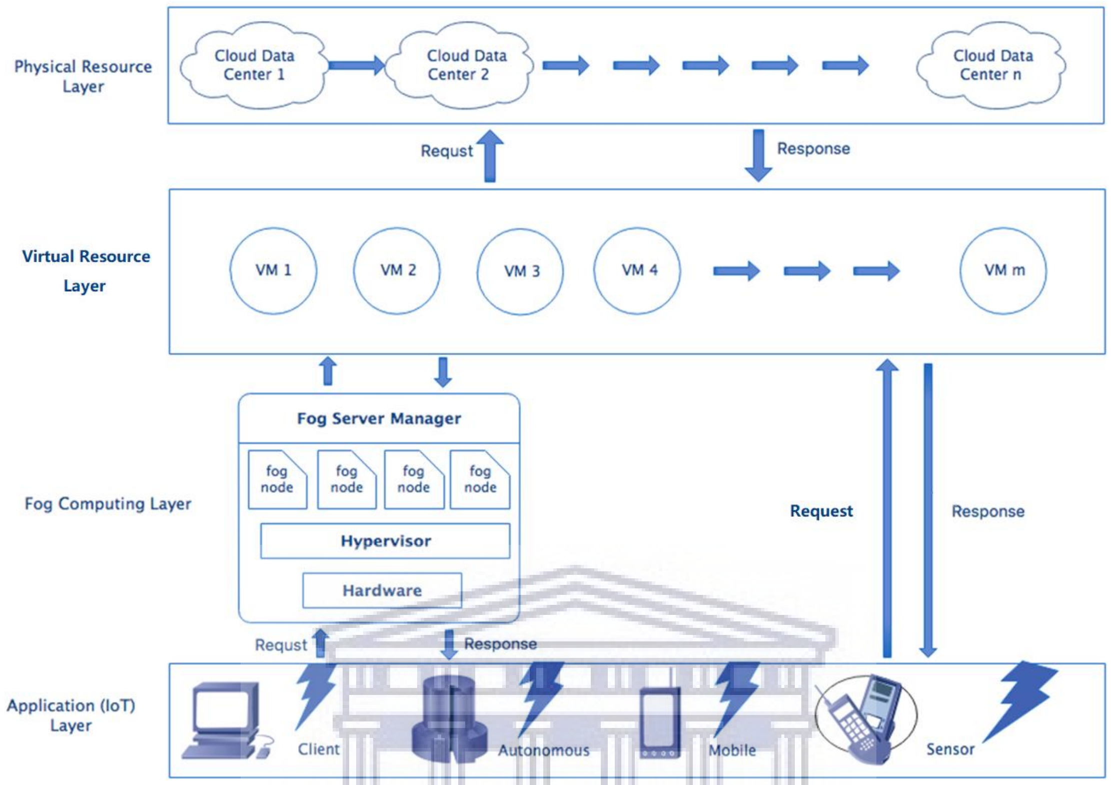


Figure 1-1 Cloud Resource layer structure

Within the Cloud Computing environment, we need to manage heterogeneous resources, such as computers, VMs, storage units, IoT platforms etc. (shown in Figure 1-1), in a cost-effective manner. In this thesis, we study the resource allocation in different modules: (i) For the creation and maintenance of physical resource layer in a collaborative Cloud federation, we proposed a fairness profit strategy based on each member's contribution (chapter 2). (ii) Between application layer and virtual resource layer, there is the task allocation management module, which includes QoS-based task scheduling. (chapter 3); (iii) Between virtual resource layer and physical resource layer, an advanced VM migration strategy is proposed in the virtual resource scheduling module to improve (chapter 4) and (iv) Between the IoT layer and Fog Computing layer, there is the edge computing-based task allocation management module, we realize the modelling and modification on both IoT layer and fog layer (chapter 5). Each setting raises different research questions and we will further discuss these questions and its challenges in the following.

(i) Formation and stability in Cloud federation with a fair profit distributing.

Cloud Computing provides a seemingly infinite infrastructure for hosting and deploying web-based applications. This enables companies or individual easily rent infrastructure resources from virtually unlimited capacity as needed. A 'pay as you go' model is used for billing, wherein users are charged based on actual resources used during a given time interval. This enables companies optimize their information technology (IT) investments, ensure resource availability and increase scalability.

While Cloud Computing offers many benefits, it has some major limitations, prominent among which are vendor lock-in and limited scalability. To overcome these limitations, the concept of a Cloud federation has been introduced. Cloud federation is a new paradigm that allows Cloud providers share resources between each other [15]. The Cloud federation can loosely be described as [16]:

- Concentration: In this federation, resource allocation is executed through a central entity. All the available Cloud resources are registered with the central entity as a repository and market for resources.
- Peer-to-peer: Here, different Cloud providers communicate directly with each other without the help of any central entity.

With respect to the Internet of Things (IoT), the ever increasing volume of “smart” or “connected” devices has unfortunately also resulted in some unwanted issues. These include: i.) increase in energy consumption of suppliers; ii.) increased latency as a result of the physical distance between IoT devices and Cloud service providers; and iii.) over utilization of Cloud provider resources. The emerging of Fog Computing and Cloud federation concepts could address these challenges, by allowing providers optimize the utilization of their resources by building business partnerships with other providers. However, the concept of balancing quality of service (QoS) with energy sustainability and cost savings is not trivial. With more and more contributions in the literature, people are paying more and more attention to this field. Currently, most energy management strategies focus on independent Cloud providers, others are beginning to focus on Cloud federation [17].

The Cloud federation could helps solve resource limitations challenges of Cloud providers, which forces them to reject new customers when there are not enough local resources to meet customer needs. The federation allows providers to dynamically outsource resources to other providers in response to changes in demand. It also allows providers that do not make full use of resources to lease some out to other providers. This outsourcing and insourcing mechanism leads to income generation and help providers get more profit when used in the right way [18]. The essence of Cloud federation is also a kind of resource allocation, but due to the selfishness of individuals, the composition and maintenance of Cloud federation also face considerable challenges [19]. Hence, the question of how to form an effective and stable federations which can attract more providers, remains a pertinent one, with the development of Cloud federations. The answer(s) to this question should address:

- Attracting new members to the federation: Cloud providers will only outsource resources or internal resources in the federation if the income earned is profitable, otherwise they will only be more willing to reject their own customer's resource request or outsource. Therefore, the resource allocation between users and Cloud providers should be effective and reasonable.
- Retaining federation members: In addition to the overall profit guarantee of the federation, the profit-sharing strategy must be distributed in a fair way among the participating Cloud providers. With this in mind, the issue of profit sharing in the Cloud federation needs to be thoroughly examined [20] and to design a dynamic and adaptive profit sharing strategy for the Cloud resource providers to guarantee each one can get his corresponding profit by his contribution.

(ii) Multiple QoS needs to be considering in task scheduling.

The essence of Cloud Computing task scheduling is a resource allocation strategy. Based on this strategy, a suitable mapping relationship between application tasks and computing resources is established to achieve reasonable allocation and efficient scheduling execution of application tasks among computing resources. However, unlike traditional distributed computing and grid computing, which only focus on performance factors such as system throughput and task completion time, the task scheduling problem in Cloud Computing environments is more complicated. First, the application task scheduling request in the Cloud Computing environment is large-scale decentralized. The Cloud Computing system must perform task scheduling and management in a distributed and parallel manner. Second, the Cloud Computing resources are often attributed to different organizations or individuals. The task scheduling strategy of the Cloud Computing system cannot interfere with the local task scheduling within its host node; further, since the Cloud Computing system is dynamically scalable, it requires that its task scheduling strategy must also be adaptive and scalable. Most importantly, as a business computing model, QoS is naturally part of the business service. In order to improve resource utilization and obtain as much service revenue as possible, Cloud Computing systems should fully consider the QoS requirements of user task scheduling. This might include addressing the following issues:

- Improving the Quality of Service of different Cloud Computing users. There are many metric for measuring of QoS in Cloud Computing, such as price, bandwidth, security, stability, etc. The goal is to design a comprehensive algorithm that considers multiple factors to achieve task scheduling based on the user's needs as well as dynamically adjust the quality content and importance.
- When signing up to a Cloud service, both users and Cloud service provider often sign a service level agreement (SLA).The SLA spells out the QoS

constraints of the user task clearly, including the deadline of the application task, the scheduling expenditure budget, the reliability of the system, and the security of the service [21]. For the relatively high network bandwidth requirements, the corresponding communication bandwidth requirements should also be agreed upon in the SLA. In order to obtain as much service revenue as possible and ensure commercial success, the Cloud Computing system must fully consider the QoS target constraint requirements of user task scheduling and meet their requirements as much as possible. Of course, in actual situations, it is impossible to ensure that all QoS are met. Therefore, according to the requirements of QoS, dynamic and adaptable task scheduling model are used to match the special QoS requirements from users.

- Optimizing the task scheduling results. The Cloud Computing system always has to deal with a large number of application tasks. The task completion time, execution cost, bandwidth support and so on refers to the compressive QoS of the whole tasks be executed by its scheduling strategy. It can be seen that when task scheduling is performed in a Cloud Computing system, maximizing QoS is a common goal of Cloud system users and Cloud service providers. Therefore, there is a need to design efficient algorithms to optimize the QoS when doing the task allocation.

(iii) **The energy based virtual resource management: VM migration and integration**.

As the number and scale of Cloud Computing data centres continue to expand, the high energy consumption of Cloud data centres has become increasingly severe. The energy-saving methods commonly used in Cloud data centres can be divided into two categories: static energy-saving methods and dynamic energy-saving methods. The mainly factors that need to be considered of the static energy-saving is designing the hardware system and its components of the computer. The method mainly includes energy-saving micro-architecture design for the motherboard [22], energy-efficient design of the circuit layer, low-power state design of the processor, memory and disk [23] and so on. The dynamic energy-saving method on the other hand, dynamically optimizes energy-saving according to the change of the operating load of the Cloud data centre from the perspective of resource management. The commonly used dynamic energy saving method mainly includes VM consolidation [5, 24-27] and Dynamic Voltage and Frequency Scaling (DVFS) [28] [29] [30]. VM consolidation is a technology for making virtual machine resource scheduling more reasonable, by means of "Live Migration" technology [27] [31]. It consolidates applications running in the Cloud data centre into a small number of compute nodes and shut down idle compute nodes to reduce energy consumption in the Cloud data centre. VM consolidation is one of the major energy-saving methods used in most Cloud Computing data centres today. In the existing research, there are still some problems worthy of further study, such as:

- How can unnecessary VM migration be avoided to conserve energy? Existing research works on virtual machine consolidation algorithms often ignore the impact of VM migration overhead when selecting VMs to be migrated. Although VMs can be quickly migrated between computing nodes within a data centre through online migration technology, VM migrations are at a cost. These cost might include reduction in running performance of VMs, increased data transfer volume in the data centre and increase in energy consumption at both root and destination compute node [5, 32, 33]. Though the cost of a single VM migration is relatively small, the wide range of tasks handled by the Cloud data centre and with the large fluctuations in the number of tasks, the Cloud data centre may experience frequent VM migration during daily operation and maintenance. These invariably results in higher overall VM migration cost has become an increasingly indispensable overhead factor in the daily management of Cloud data centres. However, when selecting the VM to be migrated, the existing research usually selects the VM to be migrated according to its resources occupied, the number of migrations, but often ignore the important factor of VM migration overhead. The proposed VM consolidation algorithm can reduce the energy consumption of the Cloud data centre to a certain extent, and also cause high migration overhead. Therefore, we need to design an efficient VM migration strategy to realize the VM migration but take into account VM migration cost as well.

- How can VM migration technology be used in federated Cloud Computing? Federation game in Cloud Computing is a way to expand/ integrate resources and make better use of resources. Each participant wanting to maximize his own interests is a key issue in the federation game. Combining VM migration and Cloud federation game must consider the different situations of the participants who wants to provide VMs. So there is a need to design the VM migration for both cooperative federation and competitive federation to meet the different requirements of participants.

(iv) The distance between Cloud Computing layer and the Internet of Things terminal layer restricts their development.

User happiness/satisfaction becomes a problem for delay-sensitive applications that require nodes to meet their latency requirements. The emerging wave of Internet deployments, especially the Internet of Things (IoT), requires mobility support and geographic distribution in addition to location awareness and low latency. A new platform is needed to meet these requirements and this is where Fog Computing comes into play [34]. Fog Computing is the infrastructure that processing power can be used from anywhere in the Cloud to the terminal equipment, it extends the power of Cloud Computing to the edge of the network, enabling any computing device to host software services and process, analyze, and store data closer to where the data was generated. For example, a Fog Computing server can process data uploaded by an IoT terminal and need to determine if it needs to be manipulated, rather than sending a meaningless data such as a temperature reading of an intelligent thermostat to the Cloud data centre every minute.

The architecture of the Fog Computing brings enormous processing power. Since its processing power is often located near the required equipment, the distance of data transmission is reduced and the delay is reduced. As a result, decisions can be made faster, and IoT manufacturers and software developers will reduce spending on Cloud Computing by limiting the amount of data sent to them. Fog Computing makes Cloud Computing better at what it does best: long-term data storage and analysis, rather than computing tasks that is time-critical. Like with the multi-layer resource allocation between Cloud Computing and its users; Fog Computing and the Internet of Things also have their corresponding challenges. Some of which include:

- How can an advanced model be built to connect IoT and Fog Computing? As the middle layer between Cloud layer and IoT layer, the fog layer plays a role of connecting the upper and lower layers. How to design a model that incorporates data collection from the IoT sensor layer and uploading same to the fog layer is a direction that needs more study.
- How can delay, distance and energy consumption between the uploaded tasks and fog nodes be balanced? In the fog layer, the fog nodes often have their own computing power and storage capacity. Therefore, the resource allocation of the Fog Computing has certain similarities with the Cloud Computing, but it pays more attention to the IoT sensor geographic location, delay time and fog nodes energy consumption of the entire system. Therefore, it is meaningful to design an efficient resource allocation algorithm to maximize the comprehensive performance of distance, delay and energy consumption for the Fog Computing layer of the model.
- How can the robustness of IoT routing protocol and battery lives of IoT

sensors be improved? In recent years, the Internet of Things (IoT), which aims to achieve universal communication between a large numbers of resources to constrain embedded devices, has become a new paradigm in the field of wireless communications. Implementing IoT basically requires thousands of low-power and low-cost embedded devices to interconnect efficiently and seamlessly [35]. Although today's routing protocols for low-power wireless networks (such as CTP or RPL) handle link failures relatively well, most studies focus on energy control, and robustness has not received enough research attention. Therefore, it is worth that we work on robustness and energy control of the routing protocol of for the equipment in IoT layer.

1.3 Contributions

(i) A unique profit sharing model for Cloud federation– Chapter 2: Though, Cloud federation can optimize resource allocation, the fairness of profit distribution between participating providers may be one of the most important conditions for joining. The economic profit distribution model of the traditional single resource level or simple proportional allocation mechanism cannot show the contribution of all participants, which means that the fair performance is insufficient. This part is to propose a fair distribution of profit, and to assess the relative importance of each participant (Cloud provider) based on the contribution of resource allocation they can make. The main contributions of this work are summarized as follows:

- Usually the proportion of participants in the Cloud federation is fixed, but in this work, a dynamic profit distribution model is considered based on the various user needs. This allows for dynamic adjustment of proportion of each participant according to different conditions.

- The proposed Cloud federation model, in addition to taking into account the time and budget conditions that users usually propose, also considers QoS attributes of reliability, stability and security of the Cloud provider. This further aids in accurately obtaining the specific computing contribution of each Cloud provider.
- Use of the economics concept of Shapley Value to calculate the contribution for participants, which can then be used for appropriate profit distribution in Cloud federation.
- Finally, the use of experiments to compare and validate the fairness of the model.

(ii) Multi-QoS constrained Cloud Computing task collaborative scheduling strategy–Chapter 3: In Cloud Computing, resource types and user preferences are heterogeneous and change dynamically. QoS constraints of user tasks are varied and with multiple metrics. The degree of satisfaction of the QoS largely determines the performance of the Cloud Computing task scheduling strategy. For the task scheduling problem with QoS target constraints in Cloud Computing environment, this chapter work proposed a multi-QoS target constrained Cloud Computing task scheduling strategy and the contributions are shown as follows:

- Proposal of an algorithm to calculate Shapley Value. Within the network of task allocation, calculating Shapley value is a NP hard problem. In this section, a modification of the algorithm proposed in [36] was proposed to calculate the Shapley value of VMs in large networks. This enabled the use of Shapley value in the federation Cloud to obtain the valuation of each member’s specific contribution.
- Modified DE algorithm based on Shapley value. The developed model is a hybrid modification of differential evolution (DE) algorithm for Cloud resource allocation with Shapley value. Specifically, the

mutation step of the DE was modified, thereby influencing the choice of genes in subsequent generations.

- Dynamic QoS Adherence: As stated above, users often have various QoS requirements that are very important to them. The proposed model is able to dynamically adapt and satisfy various user imposed QoS requirement, more specifically execution time, cost and bandwidth.

(iii) VM migration and consolidation of the virtual resource allocation – Chapter 4. Due to the existing VM consolidation research, the impact of VM migration overhead is often neglected. This part carries out the VM consolidation research of migration overhead awareness. A VM consolidation model under multiple constraints is proposed and its contributions are described as follows:

- Based on the service level agreement (SLA) violation rate and the remaining execution time of the virtual machine, we propose a VM Migration Necessity-based Dynamic Scheduling algorithm for our model. This algorithm can reduce the migration overhead during VM consolidation by selecting the virtual machine with the smallest overhead factor to migrate.
- The experimental results demonstrated the effectiveness of our proposed VM migration algorithm in VM migration times, migration mean time cost, SLA and energy consumption by comparing with many other algorithms.
- Our research also considered both cooperative federation and competitive federation while do the VM migration and the experiment results shown the performance with two different federations.

(iv) The advanced model that connects IoT and Fog Computing – Chapter 5: In recent years, the words "Internet of Things" and "Cloud Computing" have profoundly changed the IT academia and industry. However, in the actual application process, there are certain shortcomings between Internet of Things (IoT) and Cloud Computing. The IoT awareness layer has a large amount of data and is very complex, which is consist of multi-source heterogeneous data. While Cloud service is a highly aggregated service computing. Although it is cheap and convenient, it consumes a huge number of network bandwidth, and delay is also an inevitable problem due to its physical structure. In such an environment, Fog Computing merged as the times require, it is a distributed service computing model of para-virtualized architecture, which inherits the advantages of Cloud Computing and terminal computing. It can fully utilize the computing functions of the terminal and the advantages of local proximity processing. We proposed an IoT-based Fog Computing model and the contribution are described as follows:

- Mathematical modelling：Our model includes problems with the node transport protocol of the terminal layer and the allocation of resources (fog nodes) in the fog layer and tasks uploaded from the terminal layer. The purpose of this model is to minimize the overall cost of completing the terminal tasks through the fog node: time, price, energy.
- Terminal layer protocol: Based on the LIBP algorithm, we propose an improvement of the multi-sink node. The goal is to improve the robustness of the terminal layer nodes and extend the battery life of the sink node. Finally, the simulation of cooja on contiki OS proved the effectiveness of multi-sink nodes.
- Task scheduling for fog layer: A modified GA is used for optimizing the task scheduling between terminal layer and Fog Computing layer. The experiment results proof the efficient of our proposed model by

comparing with the traditional MaxMin algorithm and the fog oriented MaxMin algorithm.

1.4 Thesis Organization

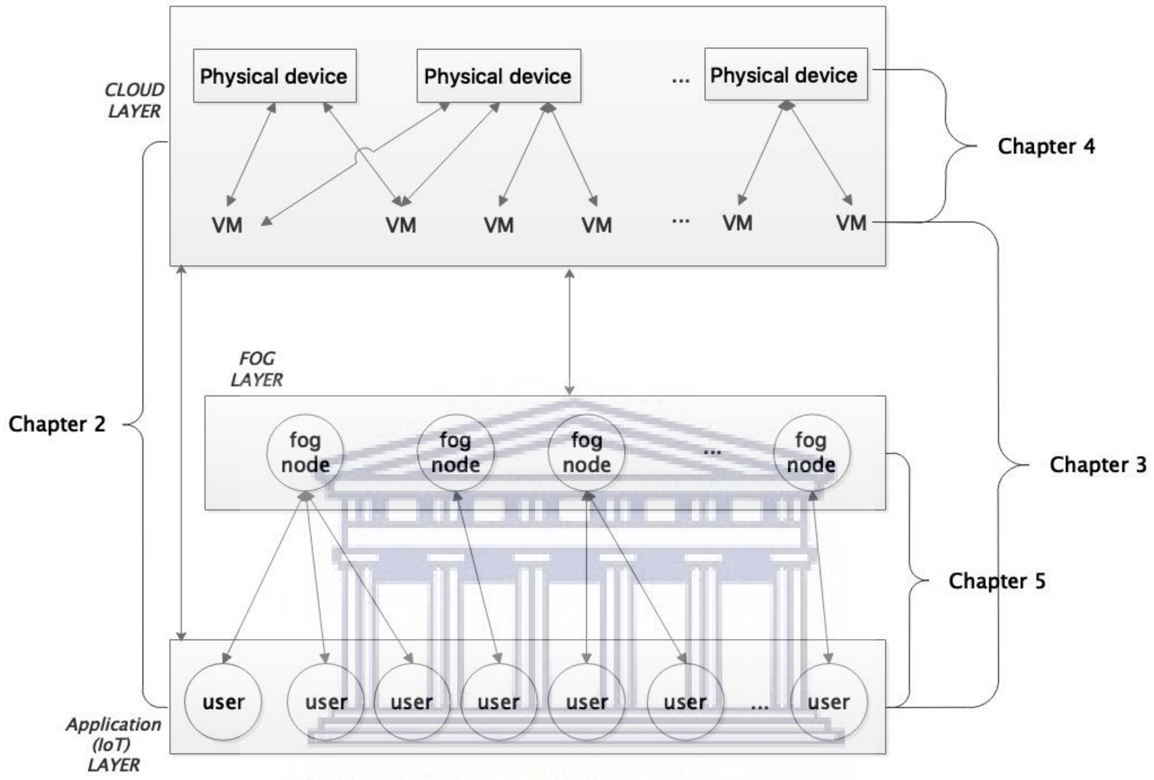


Fig 1-2 thesis constructer

As shown in the Figure 1-2, the rest of this thesis is organized as follows:

Based on the Cloud Computing environment, chapter 2 introduces the Cloud federation, analyses the characteristics of Cloud federation and the Cloud federation kernels: fairness of profit distribution and federation stability. It then discusses the proposed fairness benefit distribution strategy, which is based on the concept of economics: Shapley Value, to determine the contribution of each participant in Cloud federation. Our strategy focuses on fair and just which provides a strong support for the above two kernels. Finally, we show the validity and fairness of our strategy by the experiment.

Chapter 3 presents a Cloud Computing task collaborative scheduling strategy with multiple QoS target constraints between the virtual machines in Cloud Computing layer and users in application layer. According to the users' different task scheduling requests, the corresponding QoS target constraints are constructed respectively. Then, target solution is solved by applying the Shapley Value based Differential Evolution algorithm to optimize and dynamically adjust the QoS performance for the users' different QoS requirements.

In chapter 4, a VM consolidation model with migration overhead awareness is proposed for the VM providers in both cooperative federation and competitive federation. This model is an optimization model under multiple constraints that focuses on three factors: SLA violation rate, migration times and energy consumption. Then, the flow of the algorithm proposed in this chapter at each stage is described in detail. Finally, the algorithm was tested and verified by experiments.

For the shortcomings of communication between Cloud Computing layer and IoT layer, an IoT-based Fog Computing model is proposed in chapter 5. The model can be divided into two parts, part 1: the data processing in fog layer which focuses on optimizing the comprehensive performance of delay, distance and energy consumption. Part 2: the IoT layer which is responsible for collecting and uploading data, we proposed a modified routing protocol to enhance energy balance and nodes robustness. The sixth chapter summarizes the thesis and discusses potential future works.

Chapter 2: A Fair Profit Distribution Strategy of Modelling Cloud Federation

2.1 Introduction:

Cloud Computing is aimed at integrating IT resources into a large-scale and scalable resource pool through virtualization technology, and provides software as a Service (SAAS), Platform as a Service (PAAS) and Infrastructure as a Service (IAAS) services via the Internet [37]. Due to the dynamic nature of user requirements, especially for data-intensive needs, the increase in demand for computing resources may lead to a single Cloud Resource Provider (CRPs) being unable to meet their needs. These makes the CRPs need to improve their dynamic resource capabilities by working cooperatively as a federation. Furthermore, single Cloud Computing is associated with many other issues including: i) vendor-lock deployments which tie Cloud end-users to a unique Cloud provider, thus forbidding an average user to move its application from one Cloud to another ii) over-sized Cloud infrastructures unable to satisfy peak demand periods and leading to performance slow-down iii) non-cooperative resources and networks configuration resulting in every single service or workloads deployed in a unique site or replicated in multiple sites and iv) resources duplication resulting in departments within the same institution maintaining their own non-cooperative infrastructures.

Cloud federation [38] can make more efficient use of the Cloud infrastructure by enabling statistical multiplexing of resources and services. In many cases, this is the only or best way to satisfy the services on a global scale. The Internet is a typical example as the global service which works through a joint agreement between more than 30,000 autonomous systems or groups [39]. Currently, building the Cloud federation provides many benefits, such as capacity improvement, computationally intensive task completion time reduction, faster communication and so on.

There are two core conditions to form the federation [40]. One is maximizing the profit of the federation, which means the profit of other federation members works as any other formation won’t be higher than the federation. The other one is the profit distribution between the federation members which is the guarantee that members are willing to form the federation. If properly conceived, this strategy will provide incentives to potential members that encourage them to share their resources.

As mentioned above, based on how to define the federation game model; how to define the Cloud resource providers and Cloud federation revenue; how to define the federation members and their contributions so as to distribute the profit. This chapter present a framework for resource management for Cloud federation and propose an economic model which captures the contribution of each CRP in the user’s dynamic requisition. Based on this model, this chapter proposes the Shapley Value Profit Distribution-based Strategy (SVPDS) to distribute the profit between federation members due to each member’s contribution, that is, Shapley Value [41]. Our research reveals the following:

- Usually the proportion of participants in the Cloud federation is fixed. However, in our work, the profit distribution changes dynamically with respect to user needs. We can dynamically adjust proportion of each participate according to different conditions. The proportion of participants in the usual Cloud coalition is fixed, but in this part, the profit distribution of these participants of the federation is dynamic because of the different user demands.
- By proposing a Cloud federation model, we can take into account the time and budget conditions of users. The Quality of Service (QoS) attributes: reliability, stability and security of the Cloud provider are also considered in our model.
- Using the economic concept Shapley Value, the contribution of participants in the federation can be calculated. This provides data and strong evidence for participants profit distribution in the Cloud federation.
- Finally, the experimental results and data comparisons prove the validity and fairness of the model. More Cloud providers can participate in the distribution of

benefits, which provides strong support for the formation and consolidation of Cloud federation.

2.2 Related work

The originally conceived Cloud Computing paradigm has reached a development level, which has exposed its limitations: service disruption, degraded service quality, resource contention, data representation lack of interoperability etc. Therefore, several new methods of use and optimization have been implemented to maintain the continuity of the technology. different Cloud organizations are formed with the goal of maximizing the use of Cloud Computing, which is called inter-Clouds [42], Assis [43] compared solutions such as hybrid Cloud, Cloud and Cloud federation, they identified the functional and non-functional attributes required for Cloud federation by identifying the major architectures in the literature, and evaluate these architectures based on the attributes described.

For improving the clients experience of joining Cloud federation, Li [44] proposed a Cloud federation architecture that allows Cloud clients to seamlessly and transparently access Cloud services. The federation can be provided based on various terms, including as a subscription-based real-time online service to Cloud clients, which is aiming at simplifying the communication between clients through the Service Abstraction Layer (SAL). In order to optimize resources in heterogeneous environments and take advantage of the unlimited resources of Cloud Computing, Celesti [45] proposed a new module called Cross-Cloud Federation Manager for Cloud federation, including three agents (discovery, Match and authentication). And a technical solution based on IdP / SP model and SAML technology is proposed, which focus on the security part: authentication agent.

The Cloud federation allows the CRPs to gain more benefits through cooperation. Therefore, the Cloud federation needs to overcome the limitations of each CRP to maintain QoS during a sudden surge in resource demand. However, the presence of untrusted CRP can degrade the QoS through federated services. Trusted CRP on the other hand have high value in the federation because they can extend their resources and services to maintain the level of QoS. Thus, in order to ensure the delivery of the submitted QoS, Ray [46] proposed a broker based Cloud federation architecture. The formation of the Cloud federation is modelled as a hedonic league game. The main goal of this work is to find the most appropriate and stable federation which will maximize the satisfaction of each CRP based on the service of QoS. They proposed a federation game inspired Cloud joint formation (CGCFF) algorithm to improve the satisfaction, quality and profit for the federation.

In Cloud federation, QoS is one the non-negligible factors which reflects the overall performance of computer network, particularly the performance gotten by the users. In terms of computer network systems, QoS plays a very important role in providing high quality service, such as computing and information for the users. However, researchers in the field of various service applications have different definitions for QoS. Araban [47] and his group classify the characteristics of services into two categories, namely the internal attributes which are only related to the implementation of the service itself, and the external attributes that are associated with the environment in which the service is located, such as performance, reliability, integrity, availability. Ran [48]and his group suggested that there are five kinds of service, i.e. QoS attributes during the operation (including scalability, capacity, response time, reliability, availability, robustness, exception handling and accuracy, etc.), QoS attributes related to actual events (event integrity, etc.), QoS attributes related to deployment management (normative, support for standards, stability and change cyclicality, etc.), QoS attributes related to cost (cost, etc.), and security-related attributes (authentication, authorization, confidentiality, statistics, traceability, traceability, data encryption, non-repudiation, etc.). Researchers at IBM [49] classified service QoS into six categories: visibility, accessibility, integrity, throughput and response time, reliability, standard compliance and security.

Based on the above classifications, it can be seen that different researchers focus on different QoS attributes according to their own research domain characteristics, so it is quite challenge to provide a common QoS model for everyone. Here we propose a basic QoS model, which could provide efficient way to determine the QoS attribute and show a highly practical use value.

Based on the above classifications, it can be seen that different researchers focus on different QoS attributes according to their own research domain characteristics, so it is quite challenge to provide a common QoS model for everyone. Here this chapter proposes a basic QoS model, which could provide efficient way to determine the QoS attribute and show a highly practical use value.

QoS in Cloud services can be measured with different attributes, and the attributes used in this work are shown in Table 2-1.

Table 2-1: QoS attributes

QoS attributes	Description
Execution price(P)	The cost required for the Cloud service provider to perform a given task
Execution time(T)	The time it takes for the Cloud service provider to perform a given task
Reliable (RE)	The probability of a Cloud service running normally
Available (AVA)	The probability that a Cloud service can be successfully accessed
Security (SE)	The security level of the Cloud service

Beyond QoS, as mentioned above, a very important point of forming a federation is the strategy of benefits distribution between CRPs which must be carefully chosen. This is because the CRP will only join the federation when the income obtained is profitable, otherwise they will rather reject new user requests and keep their own resources and existing customers[50]. Since every participant in the federation is selfish, so the best profit -sharing strategy must be as fair as possible. With this in mind, profit sharing issues in the Cloud federation need to be considered critically and a profit-sharing method must be found to ensure the economic benefits of each CP belonging to the federation. Base on this theory, this chapter also focuses on proposing a fair profit sharing strategy. Comparing with the strategy in this part, there are some other common methods that people are using for sharing profit.

Zant *et al.* [20] proposed a proportional revenue sharing method that intuitively allocates the benefit ratio based on the provider's working time, but this method does not consider the source of the request and the number of virtual machines in the federation. If income incentives are not significant, this may not be attractive so as to force the CRP to reject task execution requests. Toosi *et al.* [51] proposed the Dynamic Pricing based revenue sharing strategy, in which the CRP’s revenue is calculated based on the price of the VM multiplied by the number of VMs. The disadvantage is that the source dynamic pricing may cause losses to the CP when the internal CP idle capacity is very low. Goiri [18] and his group improved the dynamic pricing strategy by add the factor alpha, which is called Pricing factor alpha based revenue sharing, its revenue is calculated by multiplying alpha based VM price with the number of VMs, as well as time duration. But in actual circumstances, the price of cooperation CRP will be different. Therefore, it is very important to decide which alpha of the CP to use, but they did not mention on what basis and how to calculate alpha. Hassan [52] *et al.* proposed the Broker’s strategically decided price-based revenue Sharing strategy, which is a kind of energy awareness resource and revenue sharing mechanism based on cooperative game theory. However, the strategy does not apply to centralized/peer-to-peer federation scince its pricing does not involve CP's consent. Tang and Chen [53] focuses on auction pricing strategy which is called double auction based revenue sharing strategy, CRPs acting as buyer and seller who respectively is presented buy bid and sell bid. A broker (the role of the auctioneer) in Cloud Federation manages all bids, performs a double auction to determine both successful buy and sale bids. The revenue sharing is dynamic changing due to these prices in different time interval. CRP can strategically manipulate bid prices and trading volumes to maximize their profits, but this strategy only works in a federation environment with brokers. Bellagio [54] and his group proposed an economic model which is a virtual currency-based auction system designed for allocating resources. Although their model provides a general approach to meeting diverse requirements, they didn’t provide a means of sharing profits among a group of independent providers whose resources are part of certain bids. Based on the dual auction of the spot market, Dramitinos [55] proposed a trading virtual machines in the certain duration. In terms of profit sharing, it is similar with the parallel synchronous markets [56, 57]. But the profit between independent organizations is implicitly shared through the market, which didn’t take into account the possibility of complementarily of users. Antoniadis [58] and his group focus on sharing the value of diversity in the federation, their research made a very fair profit distribution by Shapley Value, but they did not take into account dynamic user’s quests which may influence the contribution of members in the federation.

2.3 System Model

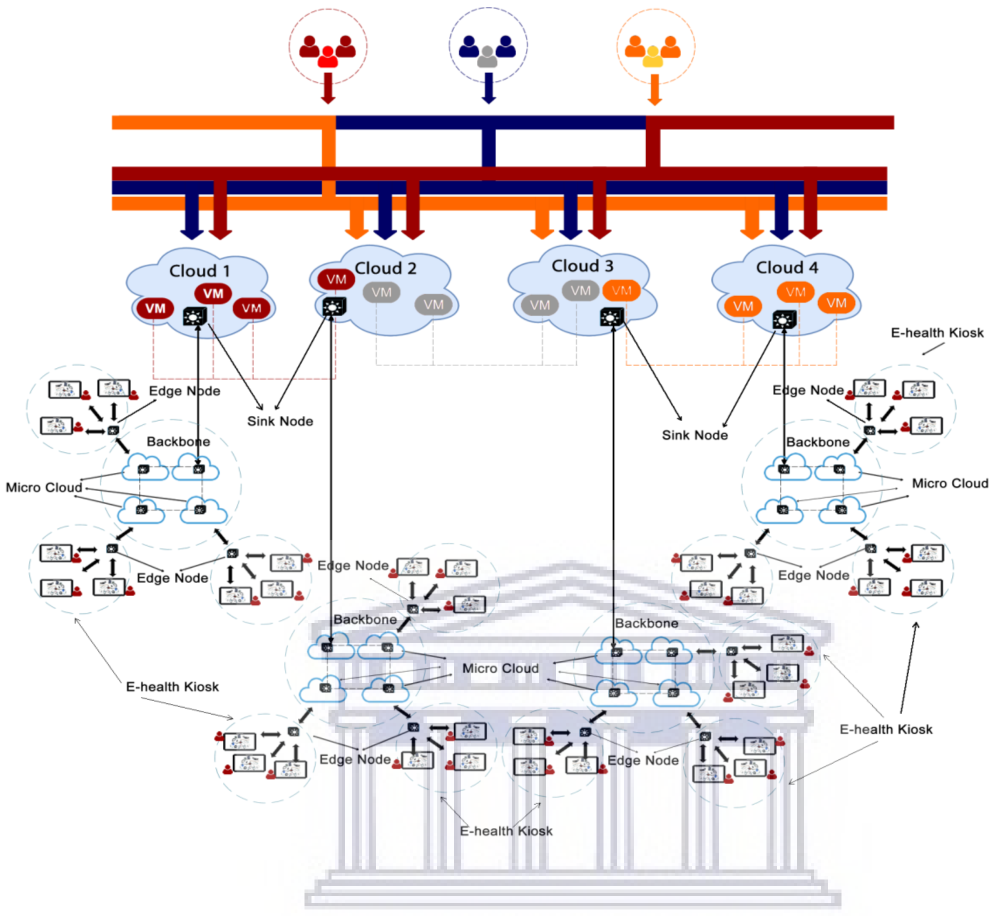


Fig. 2-1 Federated Cloud Infrastructure

A typical federated Cloud infrastructure of healthcare is depicted by Figure 2-1, it is an example of an application, where i) patients' vital signs are captured in the E-health kiosks by E-health sensors ii) routed into a network of micro-Cloud devices belonging to a fog-based infrastructure where they are aggregated and pre-processed and iii) transferred to macro-Cloud devices belonging to a local health information organization (LHIO) which are federated into a global Cloud infrastructure belonging to a regional health information organization (RHIO) sharing the Cloud resources and services. Such deployment may be suitable for rural and low-income areas of the developing countries with the expectation of enabling these settings to leapfrog from poor equipped into adequately prepared environments capable of using the federated Cloud to tackle some of the most challenges health issues of the developing world.

2.3.1. The Federated Cloud Computing Resource Management Framework

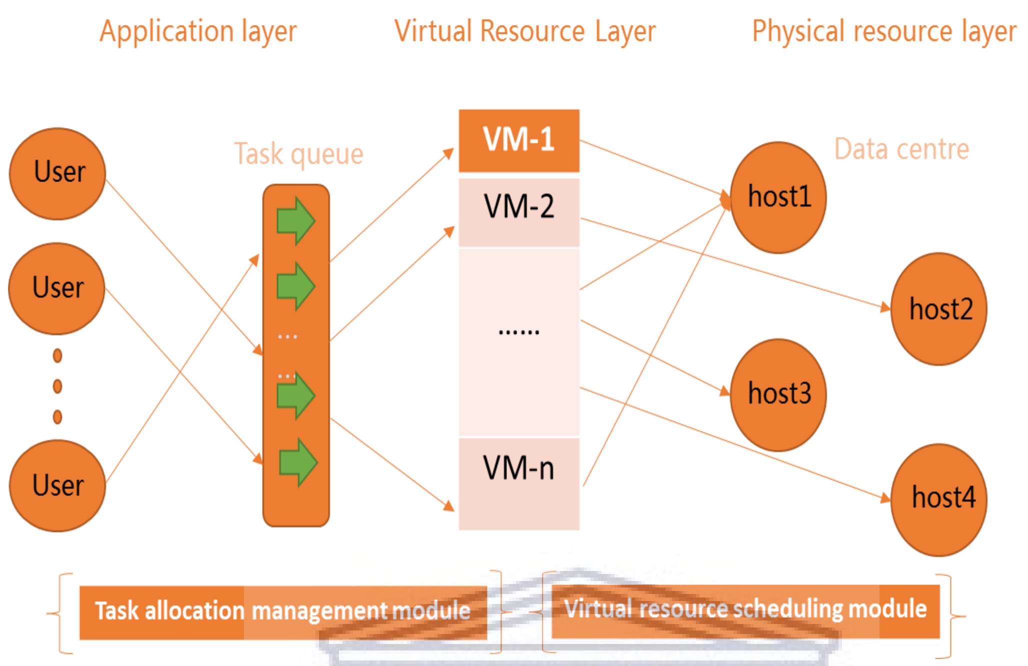


Fig. 2-2 Cloud Computing resource management framework

A Cloud Computing resource management framework is presented in Figure 2-2 where:

1. A Physical resource layer composed of data centres hosting different hardware resources as the form of VMs.
2. A virtual resource layer is layered above the physical resource layer to virtualize the physical resources for better resource management. While different virtualization techniques may be available, virtual machine virtualization and containerization seem to be the most popular techniques used by this layer to virtualize resources.
3. An application layer is layered above the virtual resource layer to provide different services to the users and include SaaS, PaaS and IaaS.

When considering a service perspective, the framework in figure 2-2 can be presented as a two-layer architecture including:

1. A virtual resource scheduling module, where a mapping between virtual and

physical resources in the data centre. Here, each physical machine host at least one virtual machine (VM) and the total performance, value, energy consumption are a summation of all VMs belong to the host who provides these VMs.

2. A task allocation management module enabling the virtual resources to be allocated to the users in a cost-effective way. Based on the structure in figure 2.2, we will discuss how to allocate the Users' task to the VMs, and how to measure the contribution of each VM which is the key of fairness distribution strategy after they finish the task execution.

2.3.2 The Cloud Federation Model

Federation game theory is an important branch of game theory, and its normalization of the interdependence of human relations is totally different from non-cooperative game. Non-cooperative game describes the model accurately to each player's action and its order, while the federation (cooperative) game focuses on the results of the formation of different coalitions of the players and the interrelationship between players and federation, i.e. the interaction between group decision makers [59, 60]. Non-cooperative game consists of four components: players, game rules, game outcomes and game effects, federation game shorten the last three elements into a whole, so the federation game is composed of two parts: one is the set of all players, the other one is the available corresponding functions of different combinations of players.

Definition 2-1 Set R = {R_1, R_2, ..., R_m} denote a collection of m Cloud resource providers (CRPs), each CRP has their own QoS attributes and also can provide resources to the user as a virtual machine instance. **VM** = {VM_1, VM_2, ..., VM_n}, and each VM instance can provide cpu, memory, bandwidth, corresponding price and so on.

Definition 2-2 Federation formation is to discuss how to divide the players Set F into disjoint federation structure. The federation structure F = {F_1, F_2, ..., F_k}

represents a federation partition form in which each player is a member of a federation that is determined. We define a specified federation we need to get its profit as S, which is F_i, $S \subseteq F$.

Definition 2-3 The Cloud providers have task set T= $\{T_1, T_2, ..., T_n\}$, where T_i is an independent task set, the deadline for completing the task set is D, and the cost budget for completing the task set is B.

Definition 2-4 The execution time function is defined as T (T, S), indicating the execution time of CRP executing task in Specified federation. We compare the execution times of all VMs, choose the longest one as the final execution time which is defined as:

$$\text{time cost:}\ \mathrm{T(T,S)} = \max \sum_{\mathrm{T\in T,CRP\in S}} \mathrm{t(T,CRP)}$$

Definition 2-5 The execution cost function is defined as P (T, S), choose the longest one as the final execution $\in \mathbf{T}$ to the resource provider $CRP_j \in S$ in Specified federation. The final execution cost is defined as:

$$\text{Execution cost:}\ \mathrm{P(T,S)} = \sum_{\mathrm{T\in T,CRP\in S}} \mathrm{p(T,CRP)}\,;$$

Definition 2-6 The reliability of the Cloud service is defined as RE: the ratio of the number of successful executions of the Cloud service to the total number of executions, that is, $RE(CRPi) = \frac{\sum_{j=1}^{n} \mathrm{REj(CR}P\mathrm{i)}}{\mathrm{n}}$, where RE_j (CRP_i) is the jth execution of the Cloud service CP_i, if the execution is successfully, then RE_j (CRP_i) is 1, otherwise it is 0.

Definition 2-7 The availability of a Cloud service is defined as AVA: the ratio of the number of successful visits to the total number of visits to the Cloud, that is, $\mathrm{AV}(CRPi) = \frac{\sum_{j=1}^{n} \mathrm{AVj(CPi)}}{\mathrm{n}}$, where AVA_j (CRP_i) is the jth visit to the Cloud service

CRP_i, if the access is successful, $AVA_j(CP_i)$ is 1, otherwise it is 0.

Definition 2-8: Cloud service security is defined as SE: refers to the Cloud service security value, if the Cloud service supports only SSL-level security, its value is 1. If the Cloud service supports WSS level Security, such as Token, Time Stamp, Signature, Encrypted, the security value is 2; otherwise, it is 0. A summary of parameters and symbols used is given on Table 2-2.

Table 2-2: Parameters and symbol definitions

Symbol	Meaning
R	The set of Cloud Resource Providers: $\{R_1, R_2, Rm\}$
F	The set of Cloud Federation which is consisted of R: $\{F_1, F_2, Fm\}$
S	One federation of the federation set F
Reliable: $RE(CRP_j)$	The probability of a Cloud service running normally which is provided by CRP_j
Available: $AVA(CRP_j)$	The probability that a Cloud service can be successfully accessed CRP_j
Security: $SE(CRP_j)$	The security level of the Cloud service CRP_j
VM	The set of VMs: $\{VM_1, VM_2, \dots VM_n\}$
Mips	The VM's CPU speed
Execution time $\mathbf{T}(\mathbf{T}, \mathbf{F})$	The cost required for the Cloud service provider to execute all tasks
Execution price $\mathbf{P}(\mathbf{T}, \mathbf{F})$	The time it takes for the Cloud service provider to execute all tasks
Task	The user's question which has to be executed T: $\{T_1, T_2, \dots T_n\}$
Budget(B)	The price that users are willing to pay
Deadline(D)	The time that users are willing to wait

Definition 2-9 the profit **V(S)** of federation S:

The goal of Cloud federation S is to provide VM instances to users while maximizing their benefits. In contrast to the work in [61], this chapter takes into account execution time, execution cost, reliability, availability and security as the feature function and defined as follows:

$$V(S)' \begin{cases} = \alpha\,(\mathrm{B} - \mathrm{P}(\mathrm{T},\mathrm{S}) + \beta\,(\mathrm{D} - \mathrm{T}(\mathrm{T},\mathrm{S})), \text{if } \mathrm{S} > 0 \; and \; T(\mathrm{T},\mathrm{S}) \le \mathrm{D} \\ = 0, \text{if } \mathrm{S} = 0 \text{ or } \mathrm{T}(\mathrm{T} < S) > D \; or \; P(T,S) > B \end{cases}$$

RAS(S)=(RE(CRP_1)+AVA(CRP_1)+SE(CRP_1))+ (RE(CRP_2)+AVA(CRP_2)+ SE(CRP_2)) +…+ (RE(CRP_n)+ AVA(CRP_n)+SE(CRP_n)).

$$V(S) \begin{cases} = V(S)' + \gamma * \mathrm{RAS}(S), \text{ if } V(S)' \neq 0 \\ = 0, \text{if } V(S)' = 0 \end{cases}$$

Where S is the federation and RAS(S) is the part of QoS attributes: reliability, availability and security of the Cloud providers. V(S)’ is the federation profit of cost and time which also belong to QoS attributes. α， β and γ are the weight factors which can be used for adjusting the importance of different QoS attribute and the V(S) is the final federation profit, v(∅)=0.

2.3.3 Federation Game

In the Cloud environment, the federation game can be defined as (S, v), and each CRP in S is a game participant, and v is the game characteristic function defined on the S ⊆ F. The characteristic function shows that the income obtained from the cooperation of CRPs in the form of a federation S is defined as v: S function⊆ F, and v (∅) = 0 Each S ⊆ F is called a federation, and if all CRPs are only one Union, known as the Grand federation, namely: S = F. In this chapter, the game feature function is defined as: $V\,(F) \begin{cases} = 0, \; |S| = 0 \\ = P, \; |S| > 0 \end{cases}$

Where: | S | represents the size of S (empty or otherwise), while P represents the total profit obtained by the federation, which is the value of the function V (S).

Cloud federation game satisfies two main properties: fairness and stability. The nature of fairness indicates that the benefits of the federation must be fairly divided among its members. Stability of the federation implies that CRP is not willing to quit the coalition. These two properties are discussed below:

1) Fairness of profit distribution

The characteristic value v(S) of federation S must be divided equally among its members based on the principle of fairness. This chapter introduces a fair sharing rule based on the market sharing mechanism, that is, the higher the contribution of resources in all possible federation in which CRP is involved, the profit it gets.

Definition 2-10 Contribution margin

CMi (S) = v (S) - v (S \ {i}), i belongs to S, represents the contribution of the player i to the federation S, where S \ {i} represents the federation formed by all the players except the player i.

Definition 2-11 Shapley value

The Shapley value measures the degree of improvement a of joining player i to the federation S by its marginal contribution. The main idea is that in federation game (N, v), where N is the federation that consist of all CRPs, player i may form a variety of different federation structure, as long as calculating the average contribution margin of all different federations of player i. Based on the contribution marginal, the Shapley value is defined as below.

In the federation game (N, v), the Shapley value represents a set of profit distributions $\Psi = (\sigma_1, \sigma_2, ..., \sigma_N)$, where σ_i=, ⊆N*P(S)*CM_i(S), P(S)=(|S|-1)!(N-|S|)!/N! means the possibility of player i forming a federation. (|S|-1)! means the number of the players before player i appears and (N-|S|)! means the number of the players after player i appears.

Definition 2-12 the profit distribution percentage: $\mathrm{PDP}(CRPi) = \frac{\Psi \mathrm{i}}{\sum_{\mathrm{j}=1}^{N} \Psi \mathrm{j}}$, where Ψ_i is the Shapley Value of CRP_i and $\sum_{\mathrm{j}=1}^{N} \Psi \mathrm{j}$ means the sum up of all members' shapley value in the federation N. The numerical analysis denote that our strategy is fairer than the profit distribution of To Each According To His Contribution (TEATHC) [62].

2) Federation stability

The following introduces the concept of kernel in the federation game to analyse the stability of the federation structure.

Definition 2-13 Distribution: A distribution that satisfies the following conditions

1) $\mathrm{PDP}(CRP_i)*V(S) \geqslant v(CRP_i)$;
2) $\sum_{\mathrm{CRPi} \in \mathrm{S}} \mathrm{PDP}(\mathrm{CRPi}) * \mathrm{V}(\mathrm{S}) = \mathrm{v}(\mathrm{S})$;

Condition 1) ensure that the profit of each member in the formation of the final federation are not less than the profit of the F_i alone, and condition 2) ensure that the total profit should be segmented among all members.

Definition 2-14 kernel is a distribution set, while satisfying $\sum_{\mathrm{CRPi} \in \mathrm{R}} \mathrm{PDP}(\mathrm{CRPi}) * \mathrm{V}(\mathrm{R}) \geq \mathrm{v}(\mathrm{S}),\ \mathrm{S} \subseteq \mathrm{F}$;

The definition of the kernel indicates that the profit of any federation is less than or equal to the sum of the dividends of its members. The existence of the profit vector in the kernel indicates that the final federation is stable. Then, if there is no arbitrary CRPs willing to leave the federation in favour of another, the profit vector is in the kernel. Please note that how to get this kernel is a NP-hard problem, usually people usually use some algorithm, such as ant colony algorithm and genetic algorithm to do the calculation. This chapter focuses on profit distribution, so we will use a simple but accurate example to illustrate and use the exhaustive method to calculate the kernel.

2.4 Algorithm Description

In this section, we set up a scenario where six Cloud tasks are to be scheduled across three CRP's Cloud resources. We analyse this example to verify the performance of the federation in completing the tasks. The parameters used are summarized on Table 2-3. Table 2-4 (a) gives the resource execution cost matrix of a task, and shows that if CRP_1 performs all tasks, the cost would be 3 + 4 + 2 + 2 + 1 + 4 = 16. Table 2-4 (b) shows that if CRP_2 performs all tasks, the task completion time would be 3 + 3 + 2 + 2 + 2 + 3 = 15.

Table 2-3: Parameter configuration

Parameter	Value
Resource providers	CRP_1, CRP_2, CRP_3
Initial federation order	$\{\{CRP_1, CRP_2\}, \{CRP_2, CRP_3\}, \{CRP_1, CRP_3\}\}$
Task Set	$T= \{T_1, T_2, T_3, T_4, T_5, T_6\}$
Budget/deadline	B=28/D=21
Weighting factor	$\alpha=0.5$, $\beta=0.5$, $\gamma1=1$
Attribute matrix	P(i,j)=Table 2(a), T(i,j)=Table 2(b), RAS(i,j)=Table 2(c)

Table 2-4(a): Attribute matrix P (i, j)

	RP_1	RP_2	RP_3
$T_1(P)$	4	3	5
$T_2(P)$	5	3	2
$T_3(P)$	3	5	4

T_4(P)	3	5	4
T_5(P)	2	5	4
T_6(P)	5	3	4

Table 2-4(b): Attribute matrix T (i, j)

	RP_1	RP_2	RP_3
T_1(T)	5	4	3
T_2(T)	3	4	6
T_3(T)	5	3	4
T_4(T)	4	3	2
T_5(T)	5	3	4
T_6(T)	5	4	3

Table 2-4(c): Attribute matrix RAS (i, j)

	reliability (RE)	availability (AVA)	security (SE)
RP_1	0.8	0.8	2
RP_2	0.9	0.7	1
RP_3	0.9	0.9	1

Algorithm 1 shows how the Shapley value can be obtained using the data on the above Tables. Note: algorithm 1 is designed to minimize the task scheduling cost.

Algorithm 1：Cost Minimizing on Time Constraint (CMTC)

```
1. input: $T = \{T_1, T_2, …, Tn\}$, $R = \{CRP_1, CRP_2, …, CRP_m\}$, Budget B, Deadline D
2. output: Mapping relation between T and R
3. Initialize Payoff matrix P (i, j), Time matrix T (i, j), k
4. For j = 1 to m
5.     Tj = 0
```

```
6. End for
7. For i = 1 to n
8.     Initialize Pmax = MAX and TP = 0
9.     For j = 1 to m
10.        If P (i, j) ≤ Pmax and TT+T(i,j)+TP ≤ D
11.            Pmax = P (i, j)
12.      TP+=Pmax
13.            K = j
14.        Else if Tj+T (i, j)>D and j==m
15.            Return infeasible allocation
16.        End if
17.        If TP>B
18.            Return infeasible allocation
19.        End if
20.      TT+=T (i, k)
21. End for return Mapping relation
```

By following algorithm 1, table 2-5 is obtained.

Table 2-5: CMTC algorithm task scheduling results

Federation structure	Scheduling results	Price	time	Revenue of cost and time	RAS value	Overall revenue
{CRP_1}	Over Time			0	0	0
{CRP_2}	Over Time			0	0	0
{CRP_3}	Over Time			0	0	0
{CRP_1, CRP_2}	$T_3,T_4,T_5 \rightarrow CRP_1$, $T_1,T_2,T_6 \rightarrow CRP_2$	17	17	7.5	3.1	10.6

{CRP_1, CRP_3}	$T_1,T_3,T_4,T_5 \rightarrow CRP_1$, $T_2,T_6 \rightarrow CRP_3$	18	21	5.5	2.6	8.1
{CRP_2, CRP_3}	$T_2,T_3,T_4,T_5 \rightarrow CRP_2$, $T_1,T_6 \rightarrow CRP_3$	20	18	5	2.7	7.7
{CRP_1, CRP_2, CRP_3}	$T_3,T_4,T_5 \rightarrow CRP_1$, $T_1,T_6 \rightarrow CRP_2$, $T_2 \rightarrow CRP_3$	16	17	8	3.2	11.2

Algorithm 1 provides a means of calculating the task scheduling results. CMTC scheduling results are shown on Table 2-5 and shows the average revenue of members. Tables 2-6 ~ 2-8 show the benefits of using the Shapley Value assignment method under the CMTC algorithm.

The revenue distribution of CRP_1 is: 0 + 10.6 / 6 + 8.1 / 6 + 3.5 / 3 = 25.7 / 6,

The revenue distribution of CRP_2 is: 0 + 10.6 / 6 + 7.7 / 6 + 3.1 / 3 = 24.5 / 6,

The revenue distribution for CRP_3 is: 0 + 8.1 / 6 + 7.7 / 6 + 1.6 / 3 = 19/6.

The Shapley value of the federation game which is solved by the CMTC algorithm is (25.7 / 6, 24.5 / 6, 19/6). This value indicates that CRP_1 contributes the most to the federation, and CRP_1 will be preferentially absorbed as federation members when building the federation. In contrast, the Shapley value of CRP_3 is minimal which means CRP_3 has the lowest priority when building federation. Therefore, the final federation structure is {{CRP_1, CRP_2}, {CRP_3}}.

Table 2-6: Revenue distribution of CRP_1 in CMTC Algorithm:

Structure	V(F)	CM(F)	|F|	P(F)	CM(F)* P(F)
{CRP_1}v	0	0	1	1/3	0

{CRP_1, CRP_2}	10.6	10.6	2	1/6	10.6/6
{CRP_1, CRP_3}	8.1	8.1	2	1/6	8.1/6
{CRP_1, CRP_2, CRP_3}	11.2	11.2-7.7=3.5	3	1/3	3.5/3

Table 2-7: Revenue distribution of CRP_2 in CMTC Algorithm:

	V(F)	CM(F)	|F|	P(F)	CM(F)* P(F)
{CRP_2}	0	0	1	1/3	0
{CRP_1, CRP_2}	10.6	10.6	2	1/6	10.6/6
{CRP_2, CRP_3}	7.7	7.7	2	1/6	7.7/6
{CRP_1, CRP_2, CRP_3}	11.2	11.2-8.1=3.1	3	1/3	3.1/3

Table 2-8: Revenue distribution of CRP_3 in CMTC Algorithm:

structure	V(F)	CM(F)	|F|	P(F)	CM(F)* P(F)
{CRP_3}	0	0	1	1/3	0
{CRP_1, CRP_3}	8.1	8.1	2	1/6	8.1/6
{CRP_2, CRP_3}	7.7	7.7	2	1/6	7.7/6
{CRP_1, CRP_2, CRP_3}	11.2	11.2-10.6=1.6	3	1/3	1.6/3

2.5 Experiment and performance analysis

In order to evaluate the proposed model, this section discusses experimental simulation carried out on Cloud-sim. The proposed model is compared with the classical Max-Min algorithm and modified Max-Min algorithm.

2.5.1 Experiment environment

Cloud-sim [63] was released by Melbourne University and Gridbus project group in 2009, which is a powerful Cloud Computing environment simulation software. It is based on the existing Java based discrete event simulation package in Grid Sim. It can also run on multiple platforms such as Windows and Linux. Our justification for choosing Cloud-sim can be described as follows:

The Cloud Computing technology is an evolution of grid technology and grid environment. Though a number of grid simulators exists, such as Grid Sim, which are good for simulating and modelling grid or distributed computing environments; they do not support the basic computing resources or any application service requirements of Cloud Computing environment [64]. The major shortcoming is their inability to model virtual resource or application tasks for the user's request according to the demand. In Cloud Computing environment, all kinds of system parameters such as the number of application tasks, the types of application tasks, the load status of the system, the level of system energy consumption, the type of available resources, the processing capacity of available resources, and the bandwidth of the network are constantly changing. The original intention of Cloud-sim simulator was to build a reasonable simulation model by this dynamic state system and the application tasks, so as to achieve the reasonable control of Cloud resources and efficient execution of application task scheduling requests. Based on Grid Sim grid simulator, Cloud-sim adds simulation support for Cloud Computing, which provides convenience for researcher on Cloud Computing technology.

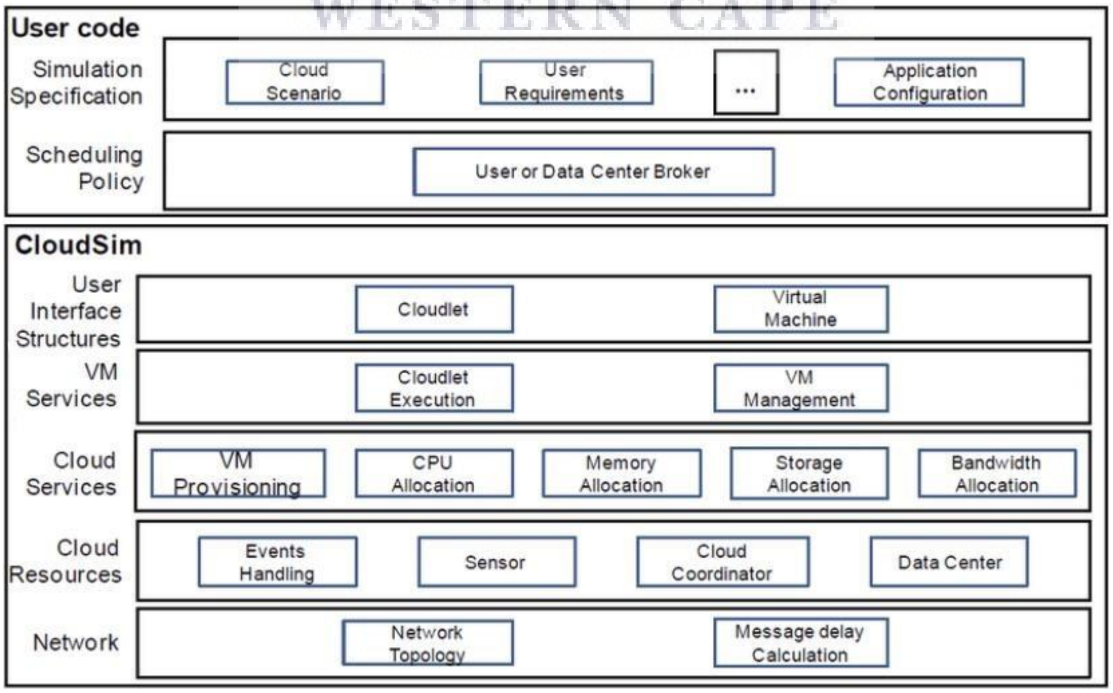


Fig. 2-3 layered architecture of Cloud Sim [63]

The architecture of the simulator is illustrated in Figure 2-3. From the figure it can be seen that the architecture of Cloud Sim is divided into four levels, namely, Sim Java, Grid Sim, Cloud Sim and User Code from bottom to top. The lowest Sim Java is the simulation engine simulator, which implements all core functions for the above three levels, such as management of various simulation modules or simulation clock, multiple components communication control, query module maintenance and so on. Based on the Java sim layer, Grid Sim layer provides a variety of software component, and also provides a graphical interface (Visual Modeller) for the users, which supports a huge convince for users. Some functional modules (the packages) which are provided by GridSim can be extended on Cloud-sim layer, to provide the computing environment simulation and simulation function of the Cloud, to instantiate the core entity and build the virtual model, and make the dynamical management at same time.

In the Cloud Sim simulator source code, the core functional modules are implemented by a series of core classes such as Datacenter, SANStorage, BWProvisioner, Memory Provisioner, VMProvisioner, Cloudlet, VMMAllocation Policy, Virtual Machine, and Datacenter Broker class and so on.

The Datacenter class is used to simulate the various core infrastructures provided by Cloud service providers in Cloud Computing systems. It encapsulates a set of computing resources, including hardware and software, and provides a series of resources allocation strategy, such as bandwidth allocation, memory allocation, and storage devices allocation. The BWProvisioner class is used to simulate the bandwidth allocation strategy of virtual machines in a Cloud data centre. This class is responsible for allocating network bandwidth resources for a set of competing virtual machines. Researchers can expand or rewrite the class as needed, to develop or test new bandwidth allocation methods. The Memory Provisioner class is used to provide a memory space allocation strategy for a group of competing virtual machines. The VMProvisioner class is responsible for selecting the host node that should be deployed for the creation request of a virtual machine in the data centre. In the default practice of Cloud Sim, this class selects the first host node that meets the virtual machine deployment requirements to deploy the current virtual machine. To create a request, in practical applications, researchers can implement better virtual machine deployment strategies by extending the VMProvisioner class. The Cloudlet class is used to simulate specific Cloud application tasks, and the specific QoS objectives of the task schedule are also set in this class. The VMAllocation Policy class is used to simulate the allocation strategy of virtual machines. Computing resources are generally divided into two types: time sharing and space sharing. Researchers can set specific resource allocation strategies by rewriting this class. The Virtual Machine class is used to simulate an instance of a specific virtual machine, and the corresponding virtual machine is managed by the host node it is deployed on. The Datacenter Broker class simulates the role of a task scheduling agent in the Cloud Computing environment.

In this chapter, to test the proposed SVPDS strategy in the Cloud Sim simulator, we first need to extend its Datacenter Broker class, implement the proposed SVPDS cooperative scheduling strategy in Datacenter Broker.java, and reload its original Bind Cloudlet to VM function.

2.5.2 Experiment setting

The operating environment of the simulation experiment in this chapter was made up of CPU: Intel (R) Core (TM) i5-2500K, 1.87GHz, Memory: 8.0GB, HDD: 1000GB.

Please notice that getting Shapley value is a NP-hard problem, so in this case we use the exhaustive method to build a simple example which is able to use the accurate data to do our experiment. Our experiment has 4 CRPs, with each CRP having 1 VM. The VM's mips, usage cost and related features are summarized on Table 2-10, and follows the Amazon EC2 on-demand instance pricing method. The experiment has 10 different requisitions, which are respectively from 1 task to 10 tasks, and we set each task length to 400. Table 2-9 shows the budget, deadline and the weight factor.

Table 2-9: Experimental parameters

Parameter	Value
Resource providers	CRP_1, CRP_2, CRP_3, CRP_4
Task Set	$T= \{T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9, T_{10}\}$
Budget/deadline	B=800/D=15
Weighting factor	α=0.5, β=0.5，γ=1
Attribute matrix	Table 2-9:

Table 2-10:

	VM0	VM1	VM2	VM3
Mips	150	140	130	80
Price	8	8	10	11
Reliability (RE)	0.3	0.5	1.0	1.0
Availability (AVA)	0.2	0.5	1.0	1.0
Security (SE)	0.5	2.0	1.0	2.0

2.5.3 Simulation results

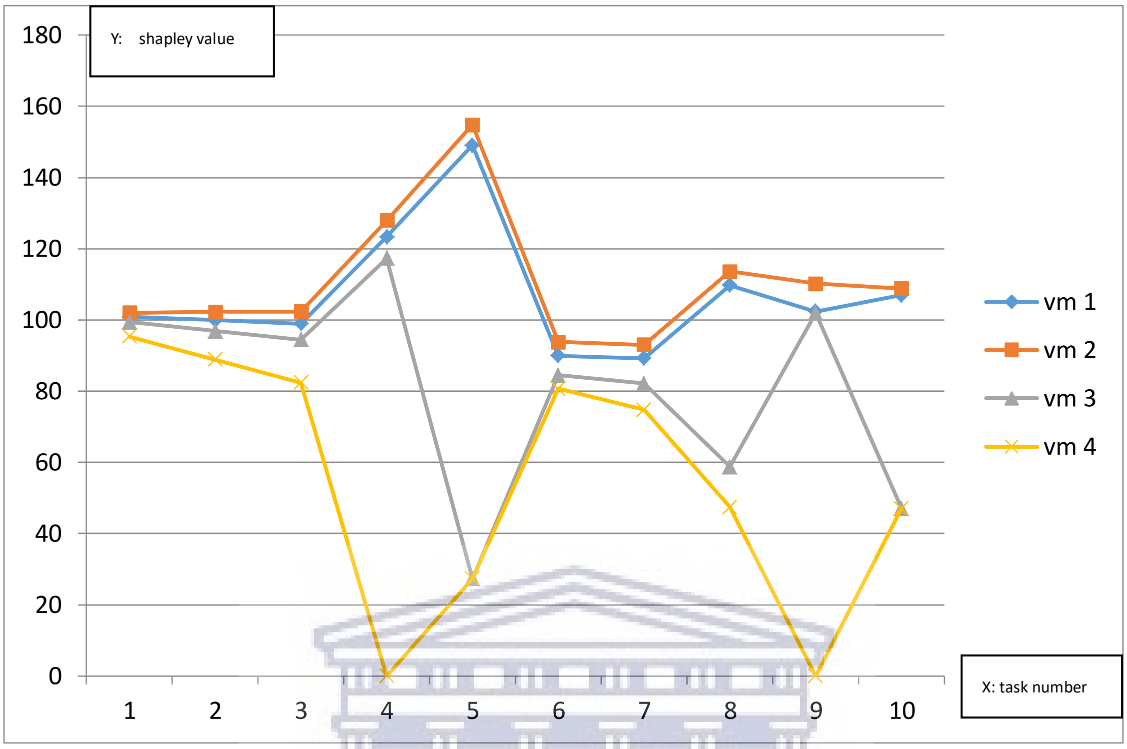


Fig. 2-4 Shapley value of each VM by SVPDS

Figure 2-4 shows a comparison of the Shapley value of each VM under different tasks. For tasks 1 to 3, it can be seen that all 4 VMs' Shapley Value were greater than 0. This means they can all get their corresponding profit in the ratio of their contributions. When the number of submitted tasks grew to 4, VM_4's Shapley Value became 0. That is because VM_4 is not able to execute 4 tasks alone with the constrain-deadline/budge, while the other 3 VMs could execute these 4 tasks alone. This implies that whatever the federation is, VM_4 won't do any execution to make any contribution.

When the number of tasks grew to 6, the Shapley Value of VM_1 and VM_2 decreased, while those of VM_3 and VM_4 increased. This means VM_1 and VM_2 were not able to execute all tasks alone, but if one of them worked with other VMs then the 6 tasks could still be finished within constrain-deadline/budget. With 9 tasks, VM_4's Shapley Value once again became 0. In essence whatever the federation is, the other

VMs do not need to work with VM_4, as without it the federation made more profit. When the number of tasks grew to 10, the Shapley Value of VM_4 increased and the Shapley Value of VM_3 decreased to the same level as VM_4. That was because both VM_3 and VM_4 were unable to make contributions alone, but if they worked as a team, they could make viable contributions.

According to definition 2-12 with the data of Figure 2-4, we can get the profit distribution percentage by SVPDS of each VM under different number of tasks. This is showed in Figure 2-5.

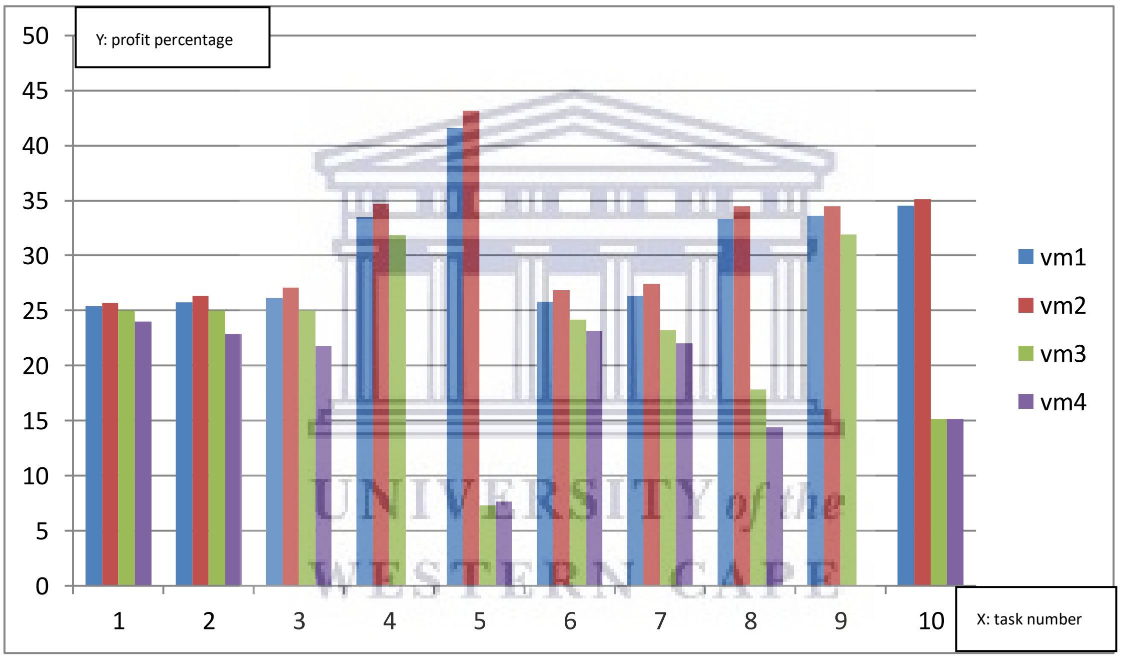


Fig. 2-5 the percentage distribution profit of each VM by Shapley Value-based Profit Distribution Strategy (SVPDS)

Through exhaustive method, we get the best task scheduling and their corresponding profit under the different number of tasks which is showed in Table 2-11.

Table 2-11: Task scheduling and profit with QoS

The	The schedule of task	Total

number of tasks		Profit
1	$Task_1$-VM_2	397.59
2	$Task_1$-VM_1, $Task_2$-VM_2	387.95
3	$Task_1$-VM_1, $Task_2$-VM_2, $Task_3$-VM_2	378.09
4	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_2, $Task_4$-VM_2	368.45
5	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_2, $Task_4$-VM_2, $Task_5$-VM_2	358.59
6	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_2, $Task_5$-VM_2, $Task_6$-VM_2	348.96
7	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_2, $Task_5$-VM_2, $Task_6$-VM_2, $Task_7$-VM_2	339.11
8	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_2, Task6-VM2, $Task_7$-VM_2, $Task_8$-VM_2	329.45
9	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_2, $Task_6$-VM_2, $Task_7$-VM_2, $Task_8$-VM_2, $Task_9$-VM_2	319.59
10	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_1, $Task_6$-VM_2, $Task_7$-VM_2, $Task_8$-VM_2, $Task_9$-VM_2, $Task_{10}$-VM_2	309.95

Using the scheduling profile on table 2-11, we get the profit distribution percentage by using TEATHC [65] for each VM under different number of tasks. This result is shown in Figure 2-6.

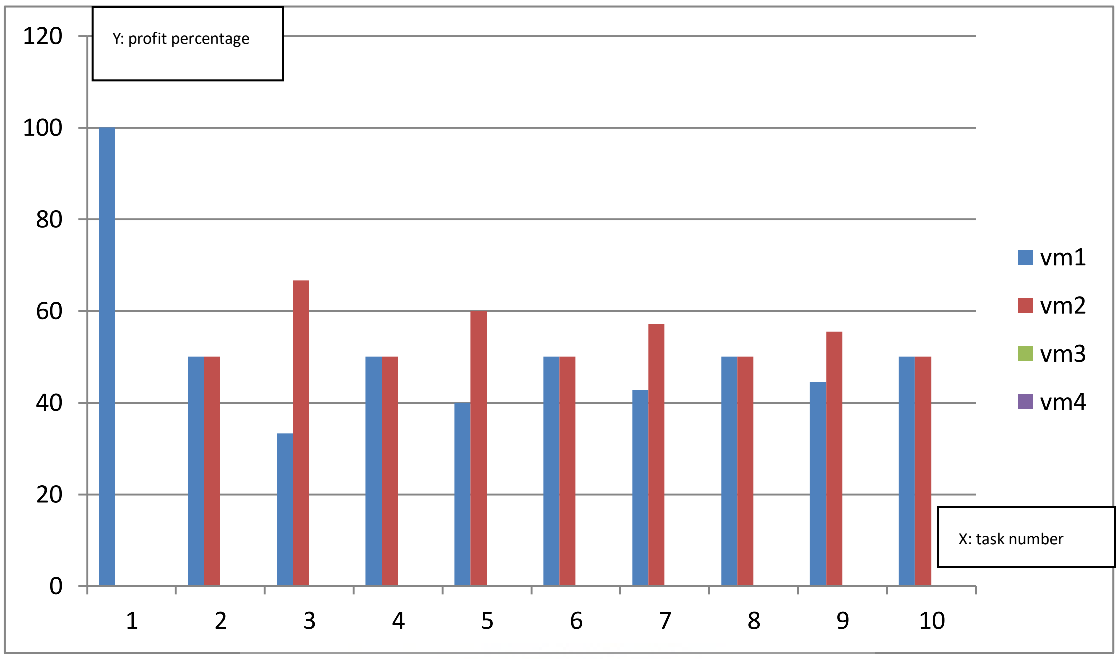


Fig. 2-6 The profit of each VM by TEATHC

Figure 2-6 shows that the VMs can be involved in 2 different profit distribution strategies. Form the task number 1 to 10, we can see in most case, only VM_1 and/or VM_2 is enough to finish tasks execution, while VM_3 and VM_4 remained on standby. This was the case when TEATHC, one of the classic tradition economic profit distribution strategies, was used. In this case its fairness is inadequate as it kept VM_3 and VM_4 running but unutilized and not profitable.

In contrast, the proposed SVPDS engaged VM_3 and VM_4 and profit was distributed fairly in accordance to their respective contributions. The comparison between these two strategies demonstrate that our proposed SVPDS is fairer. A fair benefit distribution environment not only can make the federation more stable, but also able to attract more participants to the Cloud federation.

Table 2-12: Compare SVPDS model with the TEATHC based on number of VMs involved in the profit distribution

The request	The number of tasks involve in profit distribution	
	SVPDS	TEATHC

1 task	4 VMs	1 VM
2 tasks	4 VMs	2 VMs
3 tasks	4 VMs	2 VMs
4 tasks	3 VMs	2 VMs
5 tasks	4 VMs	2 VMs
6 tasks	4 VMs	2 VMs
7 tasks	4 VMs	2 VMs
8 tasks	4 VMs	2 VMs
9 tasks	4 VMs	2 VMs
10 tasks	3 VMs	2 VMs

We conducted another experiment where QoS attributes were not considered as profit maximization constraints. Here we set the QoS weight factor γ to 0, and then obtained VMs' Shapley value and the profits under 10 different number of tasks which are respectively showed as Figure 2-7 and Table 2-13. The results without QoS attributes show that the system utilized VM_1 more than VM_2. This was because though both VMs had the same price, VM_1 was faster. This is in contrast to Table 2-11 which utilized VM_2. That is because when considering QoS, speed and price, VM_2 has more contribution to the federation than VM_1. The profits are showed in the total profit column in both Table 2-11 and Table 2-13. The above comparisons demonstrate that QoS attributes play an important role in maximizing profit for the federation.

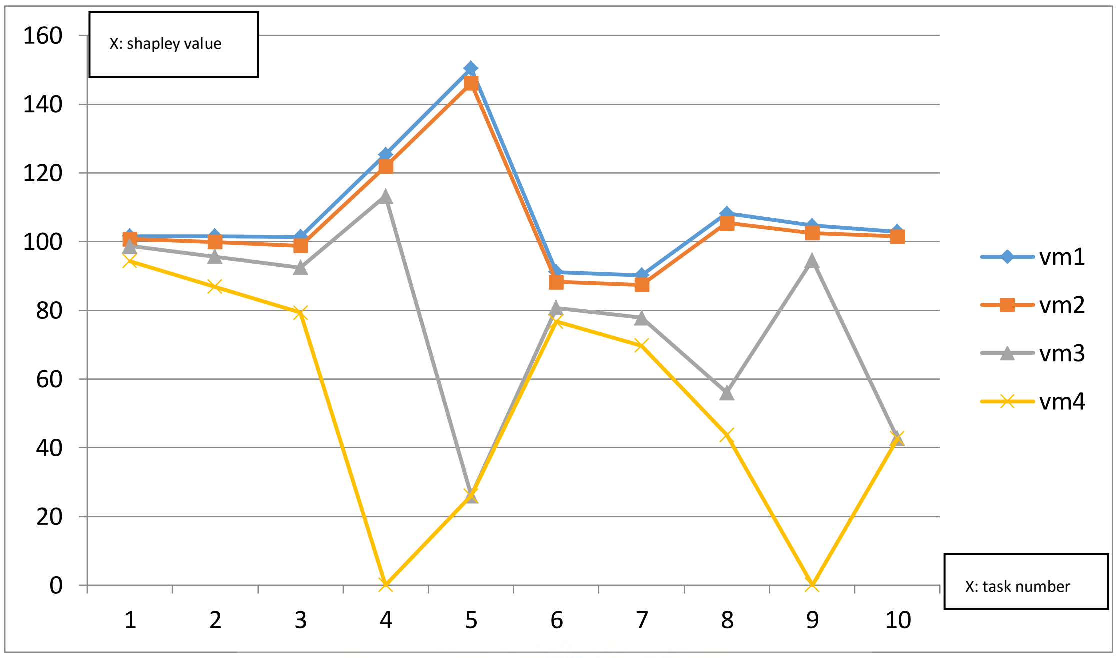


Fig.2-7 shapley value of each vm by SVPDS(γ=0)

Table 2-13 the task schedule and profit without QoS

The number of tasks	The schedule of task	Total Profit
1	$Task_1$-VM_1	396.45
2	$Task_1$-VM_1, $Task_2$-VM_2	387.95
3	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_2	377.04
4	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_2, $Task_4$-VM_2	368.45
5	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_2, $Task_5$-VM_2	357.64
6	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_2, $Task_5$-VM_2, $Task_6$-VM_2	348.96
7	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_2, $Task_6$-VM_2, $Task_7$-VM_2	338.25
8	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_2, $Task_6$-VM_2, $Task_7$-VM_2, $Task_8$-VM_2	329.45
9	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_1, $Task_6$-VM_2, $Task_7$-VM_2, $Task_8$-VM_2, $Task_9$-VM_2	318.84

10	$Task_1$-VM_1, $Task_2$-VM_1, $Task_3$-VM_1, $Task_4$-VM_1, $Task_5$-VM_1, $Task_6$-VM_2, $Task_7$-VM_2, $Task_8$-VM_2, $Task_9$-VM_2, $Task_{10}$-VM_2	309.95

2.6 CONCLUSION

Thus far, this research has only scratched the surface of this issue. We define a special QoS that can be used as a standard of measuring Cloud providers' contributions. Based on these contributions we proposed a fairness profit distribution strategy for the Cloud federation. Our simple models and examples reflect the value of a member in the federation depending on their corresponding QoS contribution. We also proposed the Shapley value-based approach to measure their contributions.

Our approach leads to quantification of intuitive notions of the value of the members according to the demand of the users. QoS is used as contribution parameter, which is the key to determine the Shapley value. We believe that our approach can easily be extended to other forms of demand patterns and can lead to economically compatible profit-sharing programs that can be applied in a sustainable Cloud resource providers federation. We measured the change in the members' Shapley Value under different numbers of tasks, and record the changes in the proportion of profit they can allocate and the number of members that can participate in the final distribution of profit which is used for the comparison of TEATHC, then we found that our strategy can allow more members to participate in the distribution of profit according to their contributions.

Other interesting directions to extend this work could in the use the loose network formulas to calculate Shapley values, as proposed by Paschalidis and Liu [66]. Federation hierarchy, composite integrated topology, market-based federation profit sharing and the performance competition in the form of Cloud facilities like Microsoft Azure can all be valuable directions for further works.

Chapter 3: A Multiple QoS-based Task scheduling model for Cloud Computing

3.1 Introduction

Cloud Computing technology is a new type of business computing service model which is developed on the basis of grid computing. Cloud Computing leverages on large number of virtualized dynamic computing resources and has become a research hotspot in the IT field. These dynamic resources can provide system users with various types of computing and data storage services. However, the system needs to coordinate distributed resources belonging to different organizations and individuals as well as provide a unified access interfaces for Cloud users. A unified interface, enables Cloud users access and use Cloud resources (through the Internet), without knowing how or which resources are allocated to them and where these resources are deployed [67]. Additionally, huge computing power can be easily aggregated thus enabling effective solution to large computationally and data-intensive application problems [68]. Despite these advantages, many challenges still exist with Cloud Computing. Some of these include the dynamic and heterogeneous nature of Cloud Computing resources, wide disparity between user workloads and effectively allocating these workloads onto Cloud resources. In solving some of these challenges, numerous research work have been done in the area of Cloud Computing task scheduling and Cloud Computing resource management [69].

When tasks are being scheduling in a Cloud Computing environment, these tasks often have various quality of service (QoS) target constraints, which the users expect the Cloud Computing system to satisfy [70]. In fact, the degree of satisfaction of QoS constraint requirements is an important factor in measuring the performance of Cloud task scheduling strategy. Therefore, in the Cloud Computing environment, the task scheduling strategy usually considers the QoS target constraint requirements of the scheduling task and the performance parameters of various available resources to achieve a reasonable match between the number of application tasks and available computing resources [71, 72].

This chapter considers three task scheduling constraints, which are: deadline, scheduling budget and bandwidth constraints, and proposes a Multi-QoS-Target Scheduling Model (M-QoS-TSM). This model assumes that the tasks to be scheduled are independent of each other. In the proposed M-QoS-TSM model, in order to meet the individualized scheduling requirements of different users for different QoS target constraints, the proposed scheduling strategy first assigns weight parameters to the various constraints, then applies a fitness function to convert the multi-QoS target constraint problem into a single-objective constraint problem. It should be noted that though only three QoS target constraint parameters are considered, without loss of generality, the Cloud task scheduling model can easily be extended to have more QoS target parameters.

Given a set of application tasks to be scheduled and a fixed set of available computing resources, the task scheduling problem is one that finds a scheduling strategy to allocate tasks to computing resources to achieve efficient tasks execution. This is an NP-complete problem, and nearly impossible to obtain the global optimal solution within polynomial time. In response to this, various heuristic optimization methods, such as genetic algorithms [73], simulated annealing methods, Tabu search, etc., are often used to approximate the optimal solution of the task scheduling problem. In the proposed M-QoS-TSM collaborative scheduling model, once the multi-QoS target constrained optimization problem has been relaxed to a single target optimization problem, after which an optimal Differential Evolution algorithm is used to obtain the approximate solution. Finally, the scheduling decision of the application task will be executed based on the solution results.

The rest of this chapter is organized as follows. First, the state-of-the-art of Cloud task scheduling problems is presented. This is followed by the definition of terms relating to the M-QoS-TSM scheduling model. An analysis of the conditions that M-QoS-TSM scheduling strategy needs to satisfy is followed by the construction of the scheduling model. The model is then used to compute an approximate solution using the optimal Differential Evolution algorithm as a binding policy between tasks and Cloud Computing resources. Finally, multiple experiments are used to verify the effectiveness of the proposed M-QoS-TSM based on average completion time, deadline violation rate, and average scheduling overhead of the application tasks.

3.2 Related works

The previous chapter had introduced some related works with respect to QoS. This chapter further discusses research works focused of QoS and task scheduling in the Cloud environment. Over the years, task scheduling of Cloud Computing has attracted a large number of experts, scholars and technicians alike [74]. Hameed [75] outlined the problem of resource allocation in Cloud Computing, summarized main techniques in the literature based on the dimension taxonomy and comprehensively sorted them by: resource adaption policy, objective function, allocation method, allocation operation, and interoperability. In order to solve the problem of optimal resource allocation, Kaikai [76] proposed a Hadoop-based Cloud manufacturing service platform which uses the revenue function for both providers and users, but they did not consider the QoS attributes of the platform. In the work of Pillai [77], a resource allocation mechanism for machines on the Cloud based on the principles of coalition formation and game theory and experiments was proposed. Experiments were used to prove the efficient of the model by comparing it with other resource allocation methods. Like the previous work QoS constraints were not considered.

Resource scheduling strategy plays an important role in determining the performance of Cloud Computing systems; however, due to the different QoS requirements, the huge number of tasks and resources, the task scheduling problem can be extremely complicated. In terms of consuming services exposed by different providers and alleviate vendor lock-in, Anastasi [78] proposed a genetic approach for Cloud Brokering. The aim was to satisfying QoS requirements of applications though finding Infrastructure-as-a-Service (IaaS) resource. However, Cloud Computing infrastructures needs to predict the cost-benefit and the corresponding QoS experienced by users. Bruneo [79] presented a stochastic reward nets (SRNs) based model to address this. Several performance metrics are defined to test the behavior of a Cloud data centre. These include: utilization, availability, waiting time, and responsiveness, so as to adjust the data parameters for the Cloud data centre. Singh [80] proposed a Cloud workload management framework. Their research is able to improve the energy consumption, execution cost and time of different Cloud workloads through K-means on the basis of weights assigned and their QoS requirements, but they did not take into account the multiple QoS constraints. A mixed-game model was built by distinguishing the task-type preference and the resource-service capability in the research of Li [81], and they proposed an evolutionary game scheduling algorithm which is based on differentiated services, by which the QoS could be improved greatly. Samanta [82]and Kumar [83] focus on using game theory to optimal the resource allocation problem of QoS constraints, in which service providers intend to solve complex parallel computing problems through the use of resources across Cloud-based networks. For the computationally intensive application task, Daniel [84] proposed a scheduling method can handle multiple application tasks in heterogeneous platform at the same time, these applications can be independent tasks, can also contain more than one bag of task. Considering QoS constraints of task scheduling, Oprescu [85] proposed a budget constrained task scheduling scheduler, which is able to deal with multiple task bags under different minimum completion time constraints, maximum budget constraints and different performance parameters. For focusing on the computation capacity in the Cloud Computing environment, Wei Yu, LinGuan Yu, LinHung Yu and Wei [86] propose a dynamic auction mechanism to solve the allocation problem. They apply a second-priced auction mechanism for improving the efficient of resource allocation. The above researches have done the works between QoS and resource allocation, but they did not consider that the multiple QoS constraints.

In addition, there are also some recent works devoted to obtaining available computing resources information to define different QoS attributes, such as the "Network Weather Service(NWS) method proposed in literature [87]". The NWS integrate resource through a variety of methods, finally constructs a prediction model for effective availability system. Randies [88] and Aslam [89] worked on the QoS constraints such as task execution time and user trust degree in the task scheduling process. The task scheduling strategies can reduce the execution time of the task to a certain extent while ensuring system load balancing.

Since the Cloud Computing task scheduling problem has certain similarities with the task scheduling in the traditional distributed environment, some classical scheduling algorithms applied in the traditional distributed environment can also be borrowed for the Cloud Computing task scheduling problem. Random load balancing algorithm (Opportunistic Load Balancing, OLB) [90]; Minimum Execution Time (MET), Freund R [91], integrated MET and MCT's Switching Algorithm; KPB (K-Percent Best) algorithm; Min-min algorithm, Wu & Shu [92]; Max-min algorithm, Maheswaran [93], Duplex algorithm, Lai [94] are some examples. The task is designed to be efficient, and can be used for Cloud Computing task scheduling problems with appropriate improvements.

In addition, since Cloud Computing task scheduling is essentially an NP-complete problem, it is impossible to find a global optimal solution within the polynomial time complexity. In order to quickly obtain a satisfactory scheduling scheme, a large number of user tasks are mapped to suitable computing resources in a short period of time. Some classical heuristic optimization algorithms are also often used in Cloud Computing task scheduling problems, such as genetic optimization algorithms genetic [95-98], ant colony optimization algorithm ant colony [99, 100], simulated annealing algorithm [101], particle swarm optimization algorithm [102-105], Tabu search algorithm [106], and a Mapping algorithm [107] and so on.

3.3 M-QoS-TSM

3.3.1 Formalization of multiple QoS target constraints

Before we introduced the contribution of VMs, there is a need to introduce the concept of federation games. The federation game is an important branch of game theory, which focuses on the interdependence of human relationships. Non-cooperative game describes the model accurately to each player's action and its order, while the federation (cooperative) game [77] focuses on the results of the formation of different coalitions of the players and the interrelationship between players and federation, i.e. the interaction between group decision makers. Non-cooperative game consists of four components: players, game rules, game outcomes and game effects; while federation game shortens the last three elements into a whole, thus composing of two parts: set of all players and corresponding functions of different combinations of players.

In our model, the federation game consists of a set of VMs; with each subset F of VM representing a federation. The value of the federation is represented by the characteristic function v (F), which means the revenue that can be obtained when the federation members work as a whole.

Definition 1 The Cloud resource provider set is represented by VM = {VM_1, VM_2, ..., VM_m}, which means m resource providers that can complete the task.

Definition 2 The federation formation is to discuss how to divide the players (VMs) into disjoint subsets. The federation structure F = {F_1, F_2, ..., F_k} represents a federation segmentation.

Definition 3 The federation game can be defined as (F_n, v), where Fn is the set of game participants and v is the value of the game players.

Definition 4 The VMs have task set T= {T_1, T_2, ..., T_s}, and the set T includes a deadline (D) of completing the tasks, and the cost budget (B) of completing the tasks.

Definition 5 The time function, defined as t: T on set T= {T_1, T_2, ..., T_s}, and the seallocating tasks to VMs. We compare the execution times of all VMs choose the longest one as the final execution time. It is defined as:

Execution Time：$\mathrm{T}(\mathrm{T},\mathrm{Fn}) = \max \sum_{\mathrm{T}\in\mathrm{T},\mathrm{VM}\in\mathrm{Fn}} \mathrm{t}(\mathrm{T},\mathrm{VM})$； (1)

Definition 6 The cost function, which is defined as c: Tution times of all VMs choose the longest one as the final execution time.rate. e tasks, and the ∈ Fn，which is defined as:

Execution cost：$\mathrm{C}(\mathrm{T},\mathrm{Fn}) = \sum_{\mathrm{T}\in\mathrm{T},\mathrm{VM}\in\mathrm{Fn}} \mathrm{c}(\mathrm{T},\mathrm{VM})$； (2)

Definition 7 The average bandwidth function, defined as B: Tndwidth functionof all VMs choose the longest one as the final execution ti(VMm ∈ Fn), which the tasks are allocated to. It is defined as:

Bandwidth：$\mathrm{B}(\mathrm{T},\mathrm{Fn}) = (\frac{1}{\mathrm{L}}) \sum_{\mathrm{T}\in\mathrm{T},\mathrm{VM}\in\mathrm{Fn}} \mathrm{b}(\mathrm{T},\mathrm{VM})$； (3)

where the L is the number of VMs in Fn.

In order to provide resources to meet tasks requirements of users, and to maximize the QoS benefits of the task implementation, VMs can form a resource federation to execute tasks. For each federation Fn, there must be a mapping between tasks and resource providers πS: T → Fn, the mapping result can be used to calculate the federation game (Fn, v), v is the equation 4 as follows:

$$\mathrm{V}(\mathrm{Fn}) \begin{cases} = \alpha\left(\mathrm{D}-\mathrm{T}(\mathrm{T},\mathrm{Fn})\right) + \beta\left(\mathrm{B}-\mathrm{C}(\mathrm{T},\mathrm{Fn})\right) + \gamma * \mathrm{B}(\mathrm{T},\mathrm{Fn}), \\ \qquad \text{if } \mathrm{T}(\mathrm{T},\mathrm{S}) \le \mathrm{D} \text{ and } \mathrm{C}(\mathrm{T},\mathrm{Fn}) \le \mathrm{B} \\ = 0, \text{if } \mathrm{S}=0 \text{ or } \mathrm{T}(\mathrm{T},\mathrm{Fn}) > \mathrm{D} \text{ or } \mathrm{P}(\mathrm{T},\mathrm{Fn}) > \mathrm{B} \end{cases} \quad (4)$$

where Fn is the member of the federation set F, D-T(T, Fn), B-C(T, Fn) and B(T, Fn) are respectively the benefits of execution time, execution cost and average bandwidth of the tasks ∈ T with the federation Fn. V(Fn) is the federation value. α, β and γ are the weight factors of execution time, execution cost and average bandwidth.

3.3.2 Model architecture

Figure 3-1 gives the architecture of the M-QoS-TSM collaborative scheduling model proposed in this chapter.

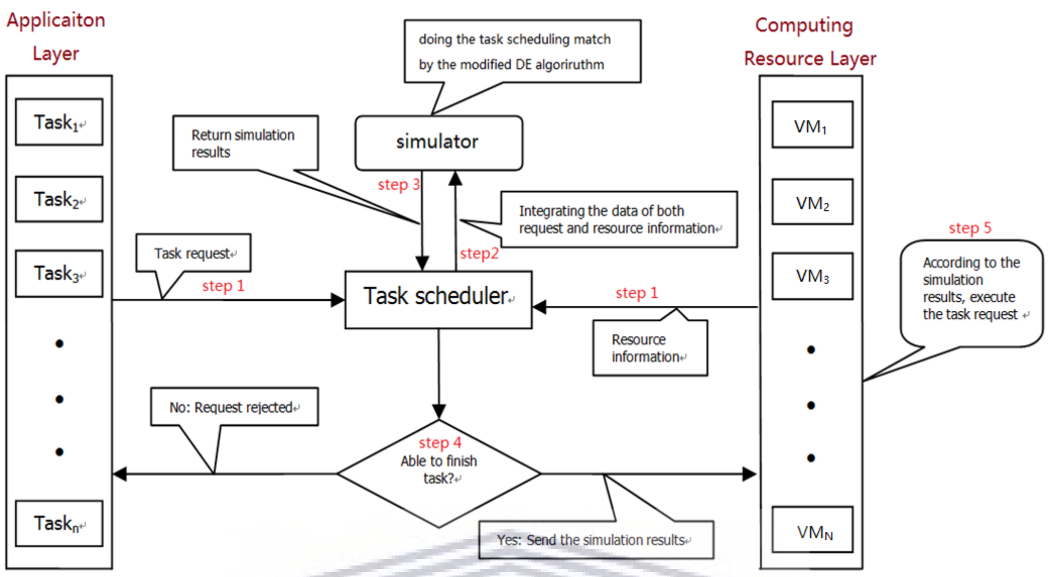

Applicaiton Layer
Task1
Task2
Task3
Taskn
doing the task scheduling match by the modified DE algoriruthm
Return simulation results
simulator
step 3
Integrating the data of both request and resource information
step2
Task request
step 1
Task scheduler
step 1
Resource information
No: Request rejected
step 4 Able to finish task?
Yes: Send the simulation results
Computing Resource Layer
VM1
VM2
VM3
VMN
step 5
According to the simulation results, execute the task request

Fig 3-1 the architecture of the M-QoS-TSM

As shown in Figure 3-1, the M-QoS-TSM scheduling model implementation process is as follows:

i. Users submit the application tasks with QoS constraints to the task scheduler, and at the same time, VMs send their resource information to the task scheduler.

ii. The scheduler transports the tasks and other related parameter information to the M-QoS-TSM model.

iii. The model generates the simulation results, then return the simulation results to task scheduler.

iv. The scheduler judges the returned information, if the VMs are able to finish the users' task request, then send the simulation result (the task scheduling message) to the VMs, otherwise it will inform the users that request has been rejected.

v. VMs execute the tasks according to the received task scheduling message.

3.4 The modified differential evolution algorithm based on Shapley Value

3.4.1 Calculating VM Contribution in Large Networks

In chapter 2, application of Shapley Value in Cloud Computing was introduced. However, though the Shapley value based centrality is superior to traditional methods, an efficient algorithm for computing its exact value for a larger network (with 100s of nodes), to the best of our knowledge, has yet to developed. For such networks, the only feasible method currently outlined in the literature is the Monte Carlo sampling [108]; this method is not only inaccurate, but also very time consuming. For example, in the network shown in [36], the Monte Carlo method must iterate 300 times, parsing the entire network a 1000 times to produce an approximation of the Shapley value, yet with an error rate of 40%. In addition, more iteration at exponential levels is needed to further reduce this margin of error.

The work of [36] considers the game defined by Suri and Narahari and proposes an accurate linear time algorithm to calculate the corresponding Shapley value of the node in a topology network. The probability that in a random permutation none of the vertices from $F(v_i) \cup \{v_j\}$ occurs before v_i, where v_i and v_j are neighbours, is $SV|v_j|=1/(1+deg(v_j))$, where $deg(v_j)$ is the number of links on each node, which is used to indicate the importance of each node (lower number of links indicate higher importance). Therefore, by summing the $SV|v_i|$ and all the $SV|v_j|$ of its neighbours, the Shapley value of the node v_i can be obtained.

In applying this model, since a VM can perform any task, the number of links of each VM is thus the same. We consider the match results between the VM and each task as the length of each link, which allows us use $deg(v_j)$ for VMs in the task scheduling. This is done in line with [36]. In practical QoS applications, users may have their own specific QoS requirements, thus an algorithm that can adjust the QoS weight dynamically is necessary. Using the different QoS requirements, the contributions can be considered as different QoS weight. The $deg(v_i)$ can be divided to three: degT(v_i), degC(v_i), degB(v_i), which respectively means the contributions of the j-th VM to the execution time , the contributions of the j-th VM to the execution cost, and j-th VM's contribution to the bandwidth. A sample example is shown in Figure 3-2, t_{11}=1, c_{11}=1, b_{11}=1 respectively means the degT(v_1), degC(v_1) and degB(v_1) of allocating $Task_1$ to VM_1, then t_{12}=2, c_{12}=2, b_{12}=2 respectively means the degT(v_1), degC(v_1) and degB(v_1) of allocating $Task_1$ to VM_2 and so on, the VM number corresponds vmSV|v_i| which can be found in algorithm 3-1.

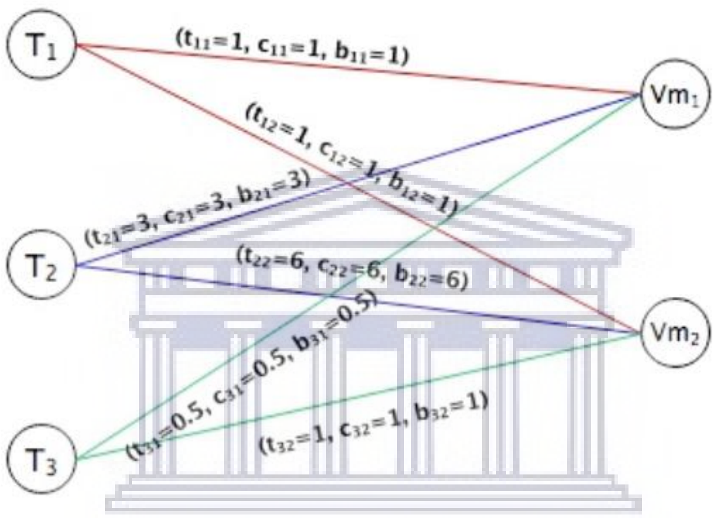

T1
T2
T3
Vm1
Vm2
(t_{11}=1, c_{11}=1, b_{11}=1)
(t_{12}=1, c_{12}=1, b_{12}=1)
(t_{21}=3, c_{21}=3, b_{21}=3)
(t_{22}=6, c_{22}=6, b_{22}=6)
(t_{31}=0.5, c_{31}=0.5, b_{31}=0.5)
(t_{32}=1, c_{32}=1, b_{32}=1)

Fig 3-2 sample example to calculate the degT, degC, degB

It is important to note that, for both execution time and execution cost smaller values are desirable, while for bandwidth, higher values are better. With respect to bandwidth, the bandwidth of any individual VM can be obtained by subtracting the bandwidth sum of all other VMs from total bandwidth (HB). This results in (7).

$$Deg(v)= \alpha /(1+degT(t))+ \beta /(1+degC(t))+ \gamma /(1+HB-degB(t)) \quad (7)$$

where α, β, γ are the weight factor to adjust the equation to match the users' QoS requirements. With deg(v), we can calculate the contribution of each VM using algorithm 1 as follows:

Algorithm 1: Computing the Shapley value for VMs

```
Input:
    virtual machine set VM, task set T
Output:
    Shapley values of all VMs
1: Initial virtual machine list: vmSV$|v|$
2: for int i=0 i$\leq$VM.length i++
3: for each $i \in Task$ do
4:     $SV|t|+=deg(vi)$;
5: end for
6: $vmSV|vi| = SV|t|$;
7: return $list : vmSV$;
```

The steps in Algorithm 1 can be summarized as follows:

- First, the set of VM and the set of T are initialized.
- Then the Shapley value of the chosen VM (vmSV|v_i|) from the set of VM is set to 0.
- Then the sum the Shapley value (SV|t|) of the chosen virtual machine with each task is determined using (7).
- Finally, the Shapley value of every chosen VM from the set of VM is put in a list (vmSV). This list provides the possibilities for the mutation step in the DE algorithm in the next section. The complexity is O(3|n||m|), where n is the number of tasks and m is the number of VMs.

3.4.2 The Modified DE algorithm

As mentioned before, task scheduling in Cloud Computing environment is a NP hard problem, hence difficult to find the best solution when large numbers of participants are involved. Intelligent optimization algorithms are usually used to find optimal / satisfactory solution. In this section, we proposed the Shapley Value based DE Algorithm (SVBDEA), which builds upon the classic DE algorithm [109]. SVBDEA requires the parameters of three main steps, population size (NP)，a parameter (S) to control the mutation, as well as the crossover probability (CR).

The steps of SVBDEA are presented as follows:

Initialing population: creating a population X consists of individuals $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}^T$, where i = 1,…,NP, and $\{X_i(0) \mid x^L_{i,j} \leq x_{i,j}(0) \leq x^U_{i,j};$ i= 1,2,…, NP; j=1,2…,E$\}$, where $x_i(0)$ is the i^th^ individual (in this thesis, individual means one match result between Tasks and VMs) and j means j^th^ dimensionality (means the number of VM), and the way to calculate $x_{i,j}(0)$ is described in equation 8 as follows:

$$x_{i,j}(0) = x^L_{i,j} + rand(0\text{-}1)*(x^U_{i,j} - x^L_{i,j}); \qquad (8)$$

where $x^L_{i,j}$, $x^U_{i,j}$ respectively express the lower bound and upper bound of j-th dimensionality, rand(0-1) means a random number between 0 and 1. In this thesis, individual x_i indicates one match result between VMs and tasks, and E is the number of tasks.

Mutation: The DE algorithm implements individual mutation through a differential strategy. The common difference strategy is to randomly select two different individuals: x_{r2} and x_{r3} in the population, then use a scaling factor (S) to scale the difference between x_{r2} and $x_{r3.}$ Than we can generate the mutant vector: $V_{i, g+1}$ by adding another random population x_{r1} according to equation 9 as follows:

$$V_{i, g+1} = x_{r1, g} + F\,(x_{r2, g} - x_{r3, g}) \qquad (9)$$

where r1, r2 and r3 are three random number in [1, NP], the S is a certain constant, and g indicates the g-th dimensionality.

The above mutation is the traditional mutation step which likes many types of mutations such as Uniform, Gaussian and Non-Uniform mutation [110]. and so on. In these mutations, the value of only a single dimensionality in the individual is changed to improve its fitness. The effect of this on the entire individual is small, especially with large population size or when the solution is close to stability. In order to improve the search range and simultaneously reduce premature convergence in a local optimal solution, we proposed using Shapley Value List: $vmSV_{i,j}$ (According the algorithm 2, vmSV|j| is the shapley value of j-th VM, j=1,2…,E; and each individual has its list: $vmSV_{i,j,}$ where i= 1,2,…, NP) to measure the importance of each dimensionality of players, to provide the possibility of each VM:: $P_{i,j}$, which can be presented in equation 10:

$$\mathrm{Pi, j} = \mathrm{vmSVi, j} / \textstyle\sum_{n=0}^{D} \mathrm{vmSVi, n} \quad (10),$$

So the $\sum_{i \in NP, j \in D} P_{i,j} = 1$, which means the sum of each $P_{i,j}$ is 100%, and the number of VM will be selected is according to applying equation 8 in the roulette wheel selection (RWS) [111], that the selected VM is defined as RWS. Since we have the possibility, we can generate new dimensionality when inheriting from individual. The mutation can be defined in equation 11 as follows:

$$x_{i,1}[n] = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}) + (z1(\triangle x_{i,1} - x_{i,1}), z2(\triangle x_{i,2} - x_{i,2}), \ldots, zn(\triangle x_{i,n} - x_{i,n})) \quad (11),$$

where z1, z2,… zn ∈ {0,1}, and $\Delta x_{i,1}, \Delta x_{i,2}, \ldots, \Delta x_{i,n}$ are selected by the RWS. Then we can generate 4 different individuals by adjusting the number of z. The first mutated individual has 1/4 dimensionalities (x) randomly set to 1, while the others are set zero. The second mutated individual has 1/2 dimensionalities randomly set to 1 and the others set to 0. The third mutated individual has 3/4 randomly set to 1 while the fourth has all its dimensionalities set to 1.

Since we have 4 mutated individuals, we then select 4 individuals with the smallest fitness value from the population and compare with the fitness values of our 4 newly mutated individuals. After the comparison, we put 4 individuals with the highest relative fitness value back into the population to get a new population.

Crossover: The purpose of crossover operation is to randomly select individuals. The method of crossover operation is shown in equation 10 as follows:

$U_{i,j}^{g+1} = V_{i,j}^{g+1}$ if rand(0,1) $\leq$ CR, j=1,…,D, (10)

where CR is the crossover probability. The crossover operation refers to optimization process. If $V(U_i^{g+1}) < V(x_i^g)$ then $x_i^g = U_i^{g+1}$, where V(·) is the fitness function of equation 4.

Description in pseudo-code of applied DE is presented in Algorithm 2.

Algorithm 2 Shapley Value-based mutation

Input:

NP: population size, ***F:*** a parameter to control the mutation, ***CR:*** crossover probability, fitness function: ***f( )*** - equation (4), a random population: x_i^G, ***generation:*** the times of generation repeat

Output:

Optimal solution (task scheduling results): individuals from last population with best fitness

```
int t = 0;
while t is not 0 do
foreach vector $x_i^G$ from population X
    do Generate mutant vector $v_i^G$ according to (7), (9), Crossover vectors according to (10)
        if $V(U_i^{g+1}) < V(x_i^g)$
        then $x_i^g = U_i^{g+1}$
        end
        **end**

    t = t + 1

    **return** Individuals of the last population with best fitness

**Stop**
```

3.5 Experiments and Performance Analysis

3.5.1 Experiment Environment

A description of the CloudSim platform and simulation environment has been given in the previous chapter. In this study, 8 VMs were used in CloudSim environment, similar to that used in [32]. Performance configuration and computing power parameters were adapted from CloudSim and are shown on Table 8.

Table 8: Performance Configuration and Computing Power Parameters (adapted from CloudSim)

:	VM1	VM 2	VM 3	VM 4	VM 5	VM 6	VM 7	VM 8
Pes	1	2	1	2	1	3	1	1
Mips	1010	2050	2130	270	550	1310	700	1180
Price	10	12	14	15	16	18	20	22
Bandwidth	100	600	210	13	100	250	60	200

In the simulator, the application task parameters include task ID and task length, in which task length use Millions of Instruction (MI) as a unit. Task length means the number of basic instructions of task scheduling requests. For this work, task lengths were set to 10000, which is adapted from the work of [33].

In the SVBDEA algorithm constructed, the parameters for the DE algorithm are set as follows: the population size of 100, maximum and minimum value are 8 and -8, the mutation zoom is 0.5, the crossover is 0.1, and the maximum iteration number of the algorithm is 2000.

3.5.3 Experiment process

The main operation steps of the experimental process are described as follows:

```
Step 1:  Grid Sim.init (num_user, calendar, trace_flag, exclude_from_file, exclude_from_proces g, report name);

/ *Cloud Sim is extended based on the core functional modules provided by Grid Sim, so the experiment needs to initialize the core module library provided by Grid Sim first. */

Step2: Data centre datacenter () = create Datacenter (" Datacenter_ 0");

/*  Create a Cloud data centre for the Cloud Computing environment, which is the resource and service provider for the entire simulation environment.*/

Step 3: Datacenter Broker datacenterbroer = create Broker ()

int broker ID = datacenterbroker.get_id ();

/* Creat task scheduling agent for Cloud Computing system, it is in charge of rescoure allocation according to the users' QoS constraints*/

Step4: virtualmachinelist = new Virtual Machine List ();

/* Create a list of virtual machines for the Cloud Computing simulation environment */

5: Step

Virtual Machine virtualmachine =
new Virtualmachine (new VMCharacteristics (vmid, bro ID, size, memory, bw, vcpus,
priority, vmm, new Time Shared VMScheduler ()));

/* Create a virtual machine and give a variety of attribute parameter settings

    Specified agent. */

Step 6: virtualmachinelist.add(virtualmachine1);

/*   Add the virtualmachine1 into the virtualmachinelist。*/

Step 7: datacenterbroker. submit VMList();

/*   Submit virtualmachinelist to   virtualmachinelistdatacenterbroker。*/

Step 8: Cloudletlist = new Cloudlet List ();

/* Creat Cloud task list*/

Step 9:

Cloudlet Cloudlet1 = new Cloudlet (id, length, file_ size, output_ size);

/* Specify the parameters for the Cloudlet */

Step10: Cloudlet1.set User ID (broker ID);

/* Assign agent of Cloudlet*/

Step11: Cloudletlist.add(Cloudlet1);

/*   Add Cloudlet1 into Cloudletlist。*/

Step12: datacenterbroker. submit Cloudlet List(Cloudletlist);

/ Submit Cloudletlist to datacenterbroker。*/
```

```
Step13: Grid Sim.start Grid Simulation ();

/* Start simulation*/

Step14: Cloudlet List newCloudlist = datacenterbroker. get Cloud List();

/*Get task list from datacenterbroker */

Step15: Grid Sim.stop Grid Simulation ();

/* Simluation finish。*/

Step16: print Cloudlet List(newCloudlist);

/* Print newCloudlist */

Step17: datacenter (). print Debts ();

/* Sum up the task scheduling overhead of users。*/
```

3.5.4 Experiment Results

In order to evaluate the proposed SVBDEA, we compared it with the usual DE algorithm [22] and the conventional task banding policy of CloudSim [24]. Due to the deadline and budget should be increased with the different number of tasks, we use the execution time and cost of conventional task allocation policy as the dynamic deadline and budget to provide the parameter for the algorithm 3. We used different values for the weight factors: α, β, γ and considered 4 different cases scenarios.

Scenario 1, α=β=γ= 1/3:

In the first scenario, we assumed that the 3 QoS attributes for users have the same degree of importance, so we set α=β=γ= 1/3. A comparison of execution times, which is a measure of the first QoS requirement is shown in Figure 3-3, The SVBDEA is on the average, respectively 22.00% and 41.13% quicker than the classic DE algorithm and the conventional task allocation policy. A comparison of the second QoS requirement (execution cost) is shown in Figure 3-4, the proposed SVBDEA results in an average of 20.83% and 39.13% less cost than the traditional DE algorithm and the conventional task allocation policy respectively. Since we are aiming at a multiple QoS objectives, it is advantageous to get higher total QoS score while sacrificing one of the QoS attributes. In this case bandwidth was sacrified, hence why SVBDEA performed worse than the other techniques in terms of bandwidth. The simulation results in Figure 3-5 shows that SVBDEA utilized on the average 6.05% and 33.70% more bandwidth than the traditional DE algorithm and the conventional task allocation policy.

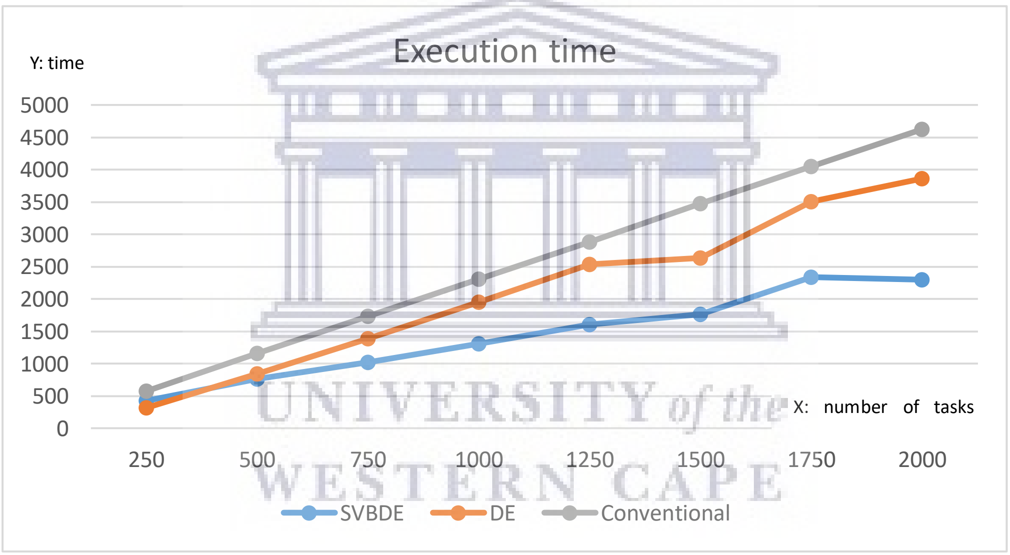


Fig. 3-3 the task execution time (makespan)

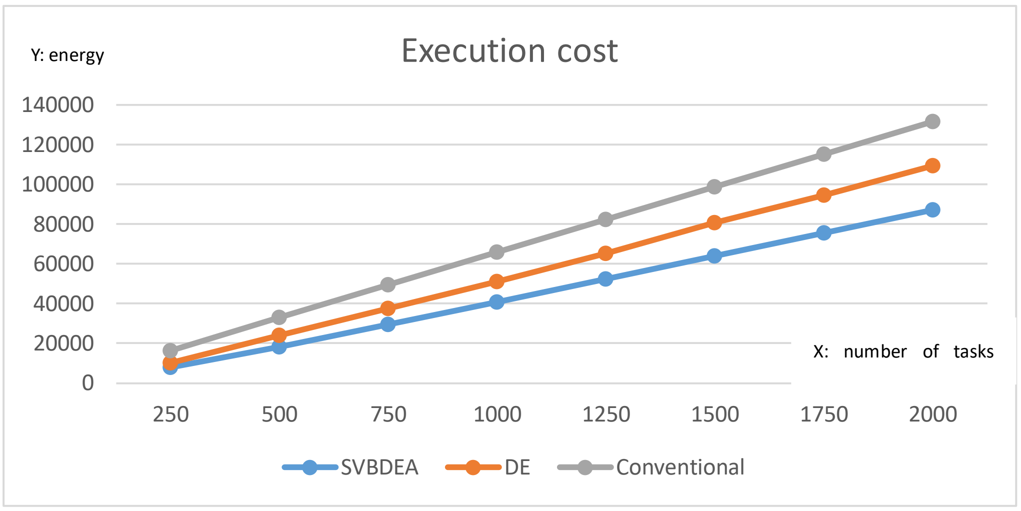


Fig. 3-4 the task execution cost

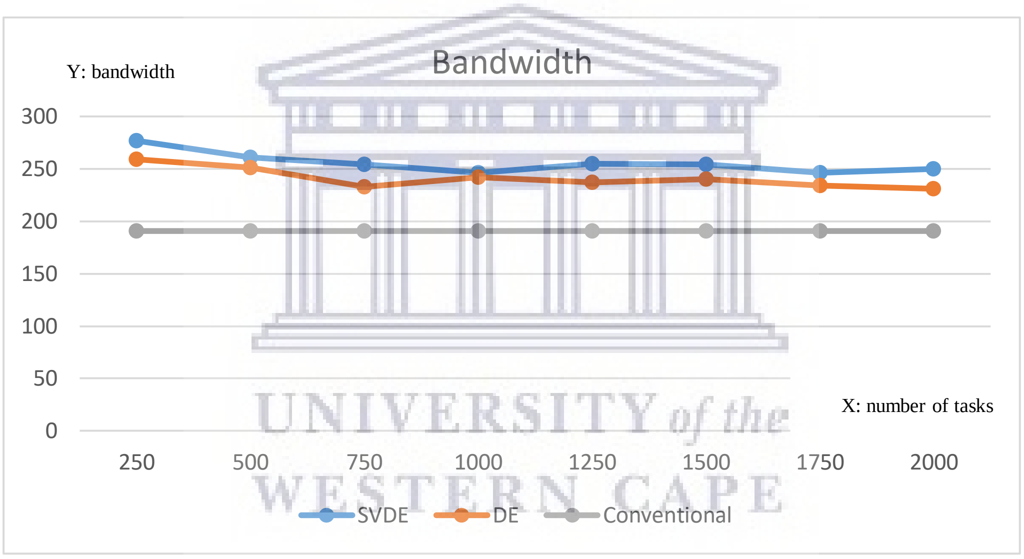


Fig. 3-5 the average bandwidth that VMs provide

In considering other QoS requirements, we considered case 2 to case 4 as follows:

Scenario 2, α= 98%, β=γ=1%, α+β+γ=100%:

In this scenario, we assumed that users only care about the execution time, so we set α to 98%, β and γ to 1%; implying that that execution time plays a more important role both than execution cost and bandwidth. The simulation results in figure 3-6 shows that SVBDEA is respectively 36.37% and 64.01% on average shorter than the usual DE algorithm and the conventional task banding policy, which is even better than the performance in case 1.

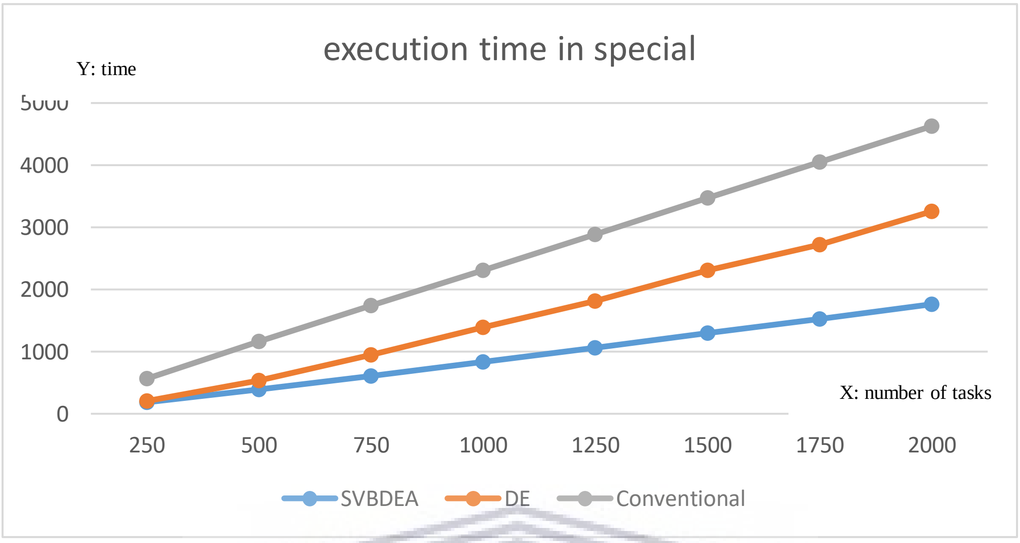


Fig. 3-6 the execution time (makespan) of special QoS requirement

Scenario 3, α=1%, β=98%, γ=1%, α+β+γ=100%:

For this scenario, we assumed that users place more importance on execution cost. We therefore, set **α** to 1%, **β** to 98% and **γ** to 1%, thereby giving execution cost higher degree of important relative to execution time and bandwidth. Obtained results are shown in Figure 3-7, on the average, SVBDEA is respectively 20.41% and 39.38% cheaper for the users than the DE algorithm and the conventional task allocation policy.

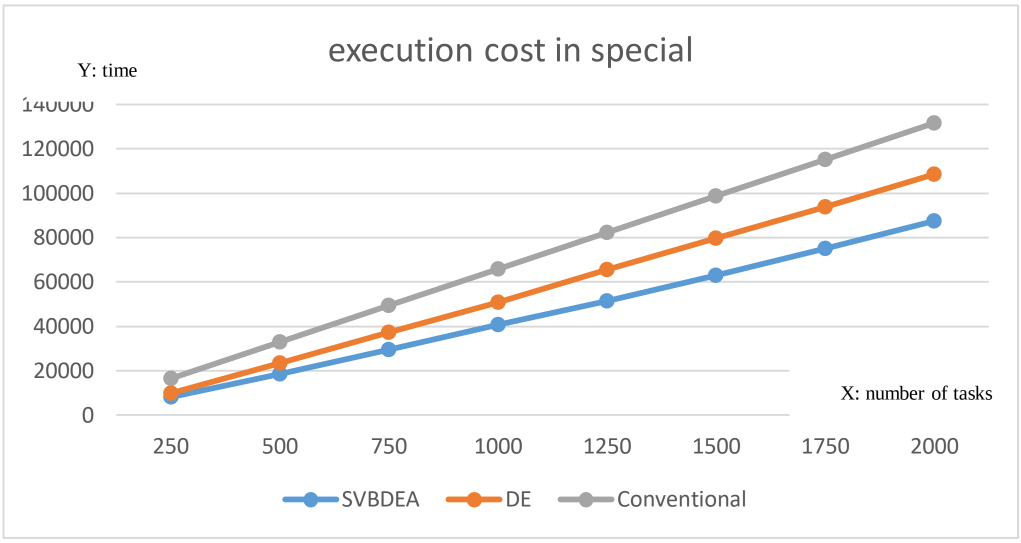


Fig. 3-7 the execution cost of special QoS requirement

Scenario 4, α=1%, β=1%, γ=98%, α+β+γ=100%:

In this scenario, we assumed that users only care about the execution time, so we set α to 1%, β to 1% and γ to 98%, which means the role that bandwidth plays is much important than execution time and execution cost. SVBDEA is respectively 8.94% and 107.79% on higher than the usual DE algorithm and the conventional task banding policy. This is as shown in Figure 3-8.

To sum up, in the task scheduling problem, the proposed SVBDEA algorithm has a better performance whether it is aiming at combination of multiple QoS or the special QoS goal.

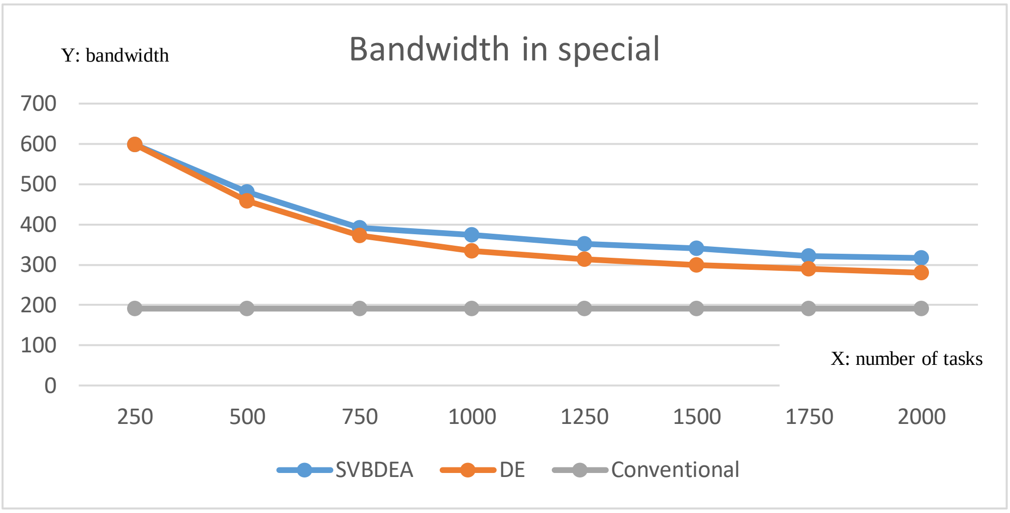


Fig. 3-8 the average bandwidth that VMs provide special QoS requirement

3.6 Conclusion

The task scheduling strategy is an important part of Cloud Computing technology. Users usually there are different constraint requirements and Cloud resources change dynamically. Reasonably and efficiently scheduling tasks to resources has thus become a major research in Cloud Computing. For the different task QoS target constraints, the scheduler must fully consider their differences and establish the reasonable mapping relationship between the application tasks and the available computing resources. In this chapter, we proposed a multi QoS target scheduling federation model (M-QoS-TSM) which considers three QoS target constraints of deadline, scheduling budget and bandwidth request. Considering that Cloud resources are fixed and we proposed a modified DE binding algorithm, which transform the multi-target constrained problem into the single target optimization problem. Though users have different task scheduling QoS constraints requests, the proposed binding policy can still satisfy these requests by adjusting the QoS constraint weight. Results of simulation show that in terms of average completion time, task bandwidth request level and the QoS performance, the model proposed in this chapter is better than the traditional DE algorithm and conventional task allocation policy in CloudSim.

Chapter 4: Migration-based VM Consolidation in Federated Cloud Computing

4.1 Introduction

Virtual machine consolidation is one of the major energy-saving methods adopted by most Cloud Computing data centres, and its technology foundation is virtual machine online migration. This technology can quickly and transparently migrate a running virtual machine from one computing node to another, which has become an important method for resource management of Cloud Computing data centres in virtualized environments. The application of virtual machine online migration can achieve the goals of load balancing, fault-tolerant management, as well as energy-saving and emission reduction in Cloud data centres [112-114]. However, during the process of migration technology, there will be a decrease of the running performance of virtual machines due to the migration, as well as an increase amount of data transmission in the data centre. These are some of the adverse effects of virtual machine migration technology [5, 32, 33, 115, 116]. In the works, these adverse effects brought by the virtual machine migration are collectively referred to as the migration cost (MC). Although the cost of a single virtual machine migration is relatively small, due to the wide range of tasks handled by the Cloud data centre and the large number of tasks being processed, Cloud virtual machine migrations may occur frequently during the daily operation and maintenance of Cloud data centres [112, 114, 117]. These make the migration of virtual machines increasingly expensive over an extended period of time. The current research on the problem of virtual machine consolidationusually select the virtual machine to be migrated according to factors such as the resources occupied by the virtual machine, the number of migrations, and ignores migration overhead can be different due to the the migration of different virtual machines. In essence, though virtual machine migration can reduce the energy consumption of Cloud data centres, it does this with high migration overheads. In addition, the remaining execution time of the virtual machine is also an important factor that affects the migration overhead. For example, when migrating a virtual machine with a short remaining execution time, for the purpose of consolidation, the corresponding migration cost still needs to be considered and this might outweigh benefit of the consolidation in itself. The migration of virtual resources in federated Cloud Computing environments is also another point, the compute nodes are from their providers, but the providers might be willing to offer their unused resources as a service to the federation (cooperative allocation) and pull back these resources for their own use when they are needed (competitive allocation). Existing research on virtual machine consolidationalso ignores this point. Therefore, in view of the above-mentioned deficiencies in the existing research, this chapter studies the impact of virtual machine migration overhead and virtual machine remaining execution time on virtual machine integration, and proposes a virtual machine consolidation algorithm with migration overhead awareness. This is adapted for both competitive Cloud federation and cooperative Cloud federation.

In this chapter, we first describe the relevant research background of the migration and migration costs of virtual machines in detail. Then, we comprehensively introduce the research status of virtual machine consolidation as well as existing selection and consolidation methods.

4.1.1Research Background

Currently, the virtual machine online migration technology generally is based on memory pre-copy migration mechanism. The migration process of a mechanism shown in Figure 4-1 [115, 118]. Before the start of pre-copy process begins, that is, before time t_0, the virtual machine (VM_1) normally runs on the source computing node. At t_0, the pre-copy operation begins, the first round of memory copying is performed, and all the memory image files occupied by VM_1 are copied from the source compute node to the destination compute node. The required time is (t_1 - t_0). Subsequently, the migration process enters the multiple rounds of memory iterative synchronization. During this round of iterations, it transfer the memory page file, which was modified by VM_1 in the previous round, from the source compute node to the destination compute node, that is, the transferred memory page size of i_{th} iteration is the portion that was modified during the (i-1)th iteration, and the modified page during the iteration process is called "Dirty Pages." The above iteration process is repeated until the size of the "dirty page" in one iteration is less than a set value or the iteration reaches a certain number of times (eg, n times). Finally, the source compute node stops the VM_1 operation at time t_n and transmits the “dirty page” generated by the n_{th} iteration to the destination compute node. After the destination compute node receives and loads the final state data, it boots up VM_1 at the time t_{n+1}. These processes are as illustrated in Figure 4-1.

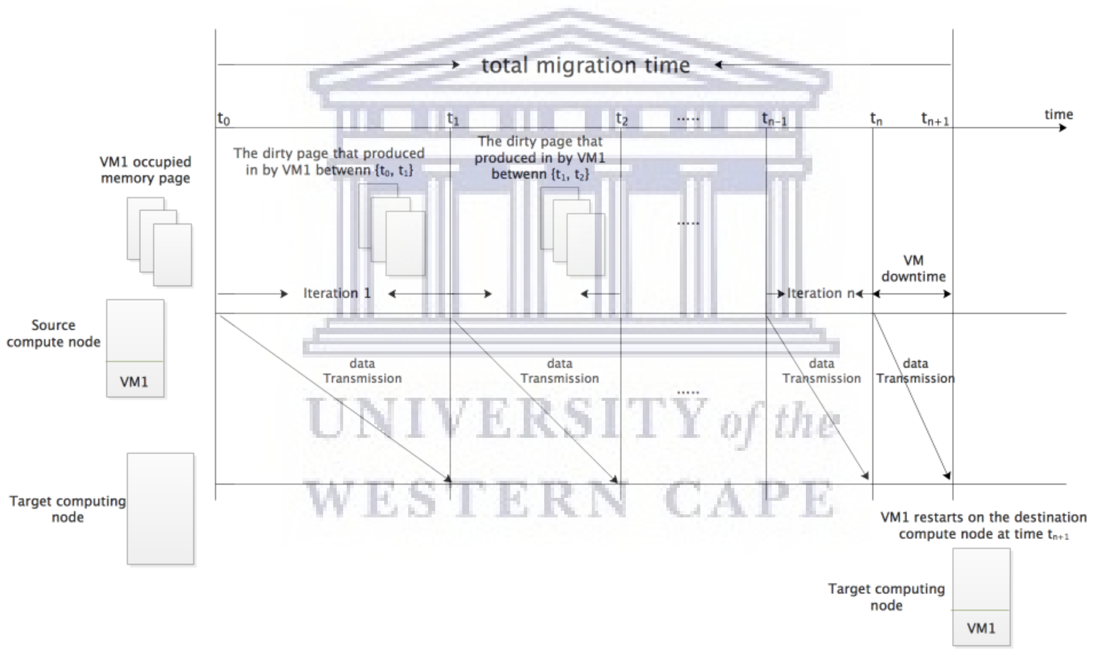


Fig 4-1 Schematic diagram of VM pre-migration process [115]

The online migration technology can realize the rapid migration of a virtual machine from a computing node within one Cloud data centre to another computing node, without sacrificing the service of this virtual machine. This greatly improves flexibility of resource management operations, such as the load balance, fault-tolerant, as well as energy conservation and emission reduction of Cloud data centre. Currently, most mainstream virtualization platforms such as VMware, Xen, KVM etc. support the application of virtual machine online migration technology. Although the process of online migration of a virtual machine is transparent to users, the slight decrease in the quality of service that occurs during the entire migration of the virtual machine is noticeable by the user. Studies have shown that the entire process of virtual machine migration will generate the following types of overhead for the Cloud data centre and the users [5, 32, 115, 116].

1) **Migration time**: refers to the length of time from the start of the pre-migration process on the source compute node to the time that virtual machine has been migrated to the destination compute node totally, as represented by T_{mig}. In the migration example in Figure 4-1, $T_{mig} = t_{n+1} - t_0$. T_{mig} is mainly affected by some factor, such as the dirty page rate of the virtual machine and the network bandwidth of the Cloud data centre. Set the memory size of the virtual machine to V_{mem}, the dirty page rate during the migration process to D_{mem}, the bandwidth of the Cloud data centre to bw, and the pre-set number of iterations to n, so that $\lambda = D_{mem} / bw$, which is provided by [115], T_{mig} can be calculated by Equation (4-1).

$$T_{mig} = \frac{V_{mem}}{bw} \cdot \frac{1-\lambda^{n+1}}{1-\lambda} \quad (4\text{-}1)$$

2) **The virtual machine downtime:** refers to the length of time that the virtual machine is neither running on the source computing node, nor running on the destination computing node, and represented by T_{down}. As shown in Figure 4-1, $T_{down} = t_{n+1} - t_n$. The migrated virtual machine is completely inaccessible during the T_{down} period, so that the downtime should be as short as possible during the virtual machine migration. Iterations number increase of of dirty page transfers is helpful for reduce the downtime, but increasing the number of iterations will also increase the migration time of the virtual machine. Therefore, the downtime and the migration time need to be considered comprehensively in the process of migration. If we define the restart time as T_{resume}, then the downtime can be described as equation (4-2).

$T_{down}=x(V_{men}/bw)+T_{resume}$ (4-2)

3) **Migration energy consumption:** expressed by E_{mig}. Virtual machine migration will increase the amount of data transmission in the Cloud data centre and occupy network resources, therefore, additional energy consumption will be generated during the migration of virtual machines [5, 32, 115].

4) **The migration performance loss:** refers to the effect of the virtual machine's service performance during the entire migration process. The study of Beloglazovet al [5] shows that during the entire migration process of the virtual machine, these is about 10% performance loss of CPU, which means the virtual machine consumes 10% less CPU resources during the entire migration process. Once a virtual machine has been migrated, its actual execution time should be increased by one-tenth of the migration time due to the 10% performance loss. The above studies on online migration of virtual machines show that although the overhead of virtual machine migration is relatively small, it cannot be ignored. The reason is that in the Cloud Computing environment, the scope of tasks handled by the data centre is relatively wide, and the number of requests submitted by users for virtual machines fluctuates greatly. In the daily operation and maintenance of Cloud data centres, frequent virtual machine migration may occur. For example, according to VMware researchers, analysis of operational profile data collected from 17 enterprise data centres who use VMware reveal that peak number of the daily operations has averaged over 3,000 times [114]. In addition, the migration costs of virtual machines are related to the types of virtual machines and the size of physical resources occupied. It can be seen that the unreasonable migration may reduce the energy consumption in the Cloud data centre but bring about a large migration overhead. In some cases, the benefits of the reduced energy consumption may be significantly less than the cost of virtual machine migration overhead. Therefore, the migration cost of virtual machines is an important factor that cannot be ignored in the process of virtual machine consolidation.

4.1.2 Related work

Virtualization technology allows Cloud providers to encapsulate the various services they provide into the form of virtual machines. When users request these services, they map these virtual machines to various computing nodes in the Cloud data centre. Through the virtual machine online migration technology, Cloud data centre managers can also dynamically change the mapping relationship between virtual machines and compute nodes according to the changes of system operational load, so as to achieve system load balancing, fault-tolerant management, and energy conservation and emission reduction goals [112, 114, 119].

Virtual machine consolidation technology can integrate compute nodes with low resource utilization by migrating the VMs that are running on them, to reduce the number of active compute nodes in Cloud data centres and achieve energy savings. In the Cloud Computing environment, the demand for resources running on the virtual machine during the execution process changes dynamically. So, the virtual machine consolidationmay cause the following situations: when the number of virtual machines running on a compute node reaches a certain level, the change of the resource demand of the virtual machine will intensify the physical resource competition, which may cause the compute node overload during the operation process. The situation is that the demand of the virtual machine running on a computing node is greater than the resource capacity provided by the computing node. Such a computing node is called a thermal computing node. At this time, it is necessary to adjust the mapping relationship between the computing node and the virtual machine by using the virtual machine online migration technology to reduce the number of virtual machines running on the thermal computing node to avoid destroying the service level agreement (SLA).

Virtual machine consolidation is often described as the Bin Packing problems. Existing studies have proposed the heuristic solutions such as First/Best Fit Descending (FFD and BFD) algorithms, and the best adaptive fallover (BAF) algorithm [120], the improved best-fit descent (MBFD) algorithm [117], the Sercon algorithm [121], and pMapper [122]. Evolutionary algorithms, such as GA, ACO, DE etc. [123-125], have also been proposed. These algorithms all provide feasible solutions for virtual machine consolidation under Cloud Computing environment. The migration necessity-based virtual machine consolidation algorithm proposed in this chapter focuses on the selection phase of the virtual machine to be migrated. It seeks to determine how to select a suitable virtual machine to be migrated when a computing node needs to migrate out virtual machines to reduce the migration overhead. Therefore, the following sections will introduce the existing research work on virtual machine selections.

The following existing virtual machine selecting method are usually adopted for virtual machine consolidation research:

(1) Random Selection Migration (RSM) [112]. The RSM method randomly selects one (or more) virtual machine from a virtual machine running on a triggering computing node to be migrated. The migration of the VM enables the previous node meet certain constraints.

(2) Resource-based Migration (Rb M) [112]. The Rb M method refers to selecting a virtual machine to be migrated according to the size of resources occupied by the virtual machine. This method can be further divided into CPU-based migration (Cb M) and memory-based migration (Mb M), and Combined Migration (Co M). The Cb M method means selecting a virtual machine that occupies a maximum (or minimum) CPU resource as the target to be migrated from the computing node. Similarly, the Mb M method is based on a memory occupation, and the Co M method is selected according to both of CPU and memory resources of the virtual machine.

(3) Balance-based Migration (Bb M) [126]. Due to the heterogeneous nature of Cloud data centre resources, different types of virtual machines may have different requirements for different resources. This method of virtual machine migration, which

makes the utilization of various resources provided by the computing nodes as consistent as possible, is called a Bb M method; conversely, the method that makes the difference in the utilization of various resources provided by the computing nodes as large as possible is called the Imbalance-based Migration (IbM) method.

(4) Minimization of Migrations (MoM) [127]. The method selects the minimum number of virtual machines to be migrated from the triggering computing nodes. This method is also one of the most adopted methods to reduce the effect of virtual machine migration.

(5) Minimum Migration Time (MMT) [5]. When a computing node in the Cloud data centre triggers a migration operation, the MMT method selects the virtual machine of the smallest migration time from the virtual machine running on the computing node. The virtual machine migration time is calculated as shown in Equation (4-1).

(6) Maximum Correlation Migration (MCM) [128-131]. This method is proposed by the following basic idea [191]: Two virtual machines running on the same computing node, the possibility of overload has a positive correlation of the resource utilization correlation of these two virtual machines. Therefore, the MCM method selects the virtual machine that has the highest CPU utilization associated with the other virtual machines running on the computing node as the target to be migrated out.

(7) Maximum Potential Growth Migration (MPGM) [117]. The potential growth of CPU utilization refers to the gap between the CPU capacity currently used by a VM and the CPU capacity set by the VM at the time of creation. The MPGM method is to select the virtual machine with the greatest potential for CPU utilization as the target to be migrated from the computing node.

Beloglazov *et al.* [117] proposed a dual-threshold based virtual machine consolidation algorithm, the author adopted three methods: RSM, MoM, and MPGM respectively to choose the virtual machine to be migrated. Finally, they compared the simulation experiments of these three methods demonstrated that the MoM method is superior to the other two methods in most cases. In [5], Beloglazov further studied the impact of virtual machine migration on the service performance, and pointed out that during the entire virtual machine migration process, the performance loss of the migrated virtual machine is equivalent to about 10% of the normal usage. In addition, the author proposes an adaptive virtual machine consolidationalgorithm and adopts RSM, MMT, and MCM methods to select the virtual machine to be migrated. In order to achieve the balance between the user's service level objective (SLO) and the system power consumption, Kord *et al.* [130] proposed a method based on the Minimum Correlation Coefficient (MCC) to place the virtual machine. Gutierrez-Garcia and Ramirez-Nafarrate [112] studied the problem of virtual machine consolidationbased on online migration, focusing on target virtual machines selecting when the computing nodes initiate migration. In order to solve these problems, the author proposes an E-protocol (Energy-aware Server Consolidation Protocol) algorithm and uses Rb M, Ib M and other methods to select the virtual machine to be migrated, and adopts a corresponding method for these virtual machines to be migrated, as well as chooses a suitable destination computing node. Mann [127] studied the placement of virtual machines in a multicore environment, in this study, the authors assume that each CPU core of a compute node can be shared by multiple virtual machine cores, so as long as the CPU is not overload, the number of virtual machine cores that each compute node can accommodate may be much larger than the number of CPUs, which improves the multiplexing capability of the compute nodes. In Mann's research, the performance of compute nodes, the number of virtual machines, and the number of overloaded CPU cores were taken into account in the process of virtual machine placement, and proposed an optimization model. The model is based on a Constraint Programming (CP) algorithm to solve the minimization problem of the model. In addition, studies such as using the MoM method to select the virtual machine which is proposed by K. S. Rao [132], is the method of Migration awareness perception (Mig

CAP) algorithm for VM migration, and its overhead is reflected in the number of VM migrations.

Based on the review above it can be seen that the most common methods used in literature to select virtual machines to be migrated are: RSM, Rb M, and MoM. Most of these methods are based on the resources occupied by virtual machines or migration times, but they often ignore the important factor of the migration cost of the virtual machine. In addition, the remaining execution time of the virtual machine is one of the important factors that need to be considered in the virtual machine consolidation process, that is always be ignored either. Therefore, aiming at the above two problems existing in the research on the consolidation of existing virtual machines, this chapter proposes a VM Migration Necessity-based Dynamic Scheduling Algorithm (VMMNDSA).

Table 4-1 Summary of Parameters and Symbols

symbol	description
T= {1，2，3,…,|T|}	T is the length of the entire run cycle, divided into |T| time windows
CN={cn$_j$,1={cnh	CN is the set of compute nodes in the data centre, where M represents the total number of compute nodes and cn$_j$ represents the jth compute node
$\Omega_j, \Gamma_j, \Lambda_j$	The CPU, memory and bandwidth resource capabilities of the compute node cn$_j$
CN(t), m(t)	The set of compute nodes used in the tth time window, and the size of the set
V(t), n(t)	The set of virtual machines used in the tth time window, and the size of the set
at(v_i), rt(v_i)	The time that v_i arrived and its execution time
ω_{it}, γ_{it}, λ_{it}	The CPU, memory and bandwidth requirements of v_i in the tth time window
$T_{mig}(v_j)$	The migration time of v_j
$T_{down}(v_j)$	The downtime of v_j
$E_{mig}(v_j)$	The migration energy consumption of v_j
$CF_{mig}(v_j)$	The migration cost of v_j
S(t)=(s_1(t), s_2(t)...,	the state vector of compute node set, where s_j(t) represents the state of cn$_j$

$s_M(t))$	
$SLA_j(t)$	Whether the compute node cn_j has an SLA violation in the t^{th} time window
$SUMT_{mig}(cn_j)$	The sum up of all virtual machine migration downtime in the compute node cn_j
LThreshold	Lower threshold for resource utilization

4.2 VM Migration Necessity-based Dynamic Scheduling Model

In this chapter, we first describe the above-mentioned migration overhead of virtual machine consolidation issues in detail, and then establish a mathematical model of the problem, including objective functions and constraints. For the convenience of reading, table 4-1 gives a summary of the parameters used in this chapter.

4.2.1 Problem Description

In order to meet the needs of users in different geographic locations, Cloud providers deploy a large number of Cloud Computing data centres around the world. These Cloud data centres can receive virtual machine requests submitted by users, and based on the different needs and the resource capability to place the received virtual machine requests on the appropriate compute nodes for task execution. However, in Cloud Computing environment, computing nodes in a Cloud data centre may be affected by various factors such as the arrival of a virtual machine request, the execution of a virtual machine request, and the dynamic changes in resource requirements during execution of the virtual machine. The resource utilization could be too low or too high, so that the VMs running on the Cloud data centre need to be dynamically adjust by VM migration. Most of the current researches on virtual machine consolidation focus on the single consolidation under the current operating state of Cloud data centres. The research in this chapter is different from these studies in that it focuses on the necessity and overhead associated with migrating individual VM. It takes into account the VM’s dynamic nature, that is, the virtual machine running to the end, the arrival of new virtual machine requests and the changes in resource requirements on the running virtual machine. This helps prevent unnecessary VM migration and consolidation, in order to achieve the Cloud data adjustment. This work also takes Cloud federation (described in chapter 2) into account. We consider that PM providers as the participants who form the federation as illustrated in Figure 4-2. In cooperative federation, VMs can be migrated to any PMs, for example in Figure 4-2, VM_1 can be migrated to PM_3. However, in competitive federation, VM_1 only can be migrated to the PM_2, which is from the same PM provider: PM provider 1.

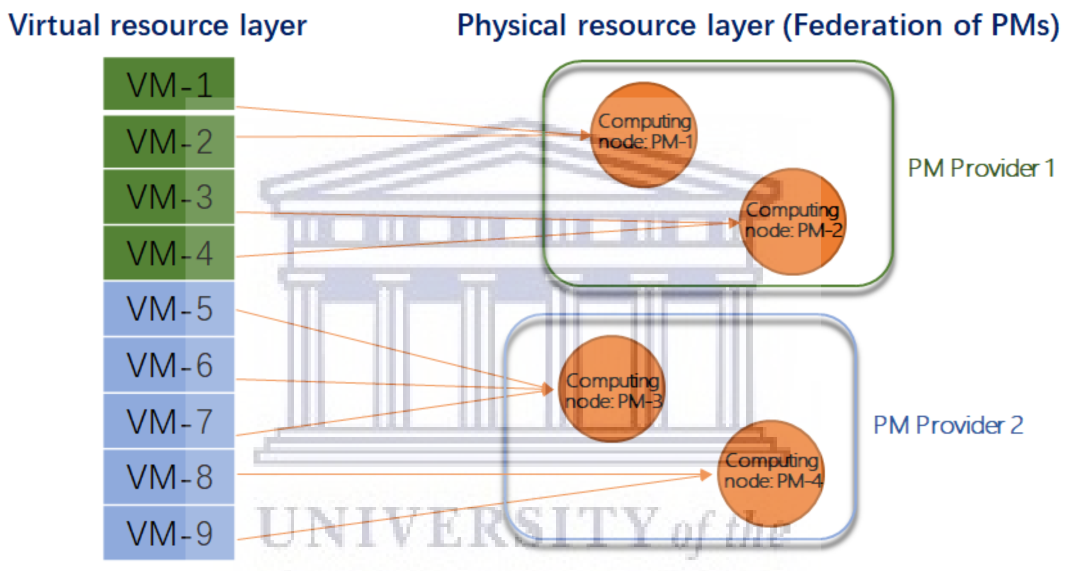


Fig 4-2 VM migration model of federated PMs in Cloud Computing

In order to more intuitively and clearly describe the virtual machine consolidation problem of migration overhead, we first give some definitions of related concepts and terms.

Definition 4-1: Time Slot (TS) [133]. Assuming that the entire data centre's running time is T, we can use the method shown in Figure 4-3 to divide the entire running time period T by the fixed unit time length tΔ to |T | unit time lengths, ie T= {1, 2, 3, ..., |T|}. A unit time length t_Δ is a time window and can be considered as a basic time processing unit. For example, the initial deployment of a virtual machine is based on a uniform placement for each time window, which is, put all the arrived VM requests into the waiting queue during the time t ($1 \leqslant t \leqslant |T|$). Then, at the beginning of the (t + 1)th time window, make the uniformly placement of all the VM requests in the waiting queue. In the experiment in this chapter, the value of the unit time length t_Δ is set to 1 minute.

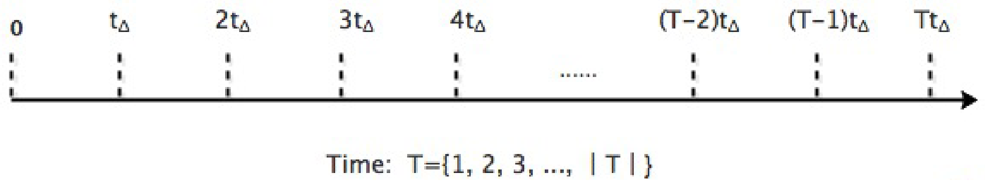


Fig.4-3 Time division window schematic

Definition 4-2: Cloud Data centre (DC). The CN={cn_j|1(DC). The CN={cnndow schematic In the experiment in this chaCloud data centre, where M is the total number of computing nodes and cn_j is the jth computing node. Due to these compute nodes are heterogeneous, their CPU processing power and memory size may different, so using Ω_j, Γ_j, and Λ_j to represent the CPU processing power, memory size and bandwidth that the compute node cn_j can provide. CN(t) denotes the set of compute nodes that the Cloud data centre is using within the tth time window, and m(t) denotes the number of compute nodes in the set CN(t). CN(t) and m(t) satisfy the following relational equations: $CN(t) \subseteq CN$ and $m(t) \leqslant M$.

Definition 4-3: Virtual Machine (VM). If V represents the set of all virtual machine requests that arrived during the entire operating cycle T of the Cloud data centre. N=|V| indicates the total number of VM requests processed by the Cloud data centre. V(t) denote the set of VMs running in the Cloud data centre at the beginning of the tth time window, then $V(t) \subseteq V$. If n(t)=|V(t)| to represent the number of virtual machines running in the Cloud DC in the tth time window and use v_i to represent the ith virtual machine in V(t), then $v_i \in v(t)$ and 1ndtual . Using $at(v_i)$, $rt(v_i)$ to respectively represents the arrival time and execution time of the VM v_i, V(t) can be described formally: $V(t)=\{vi|at(vi)<t \text{ and } at(vi)+rt(vi)>t\}$, where $at(vi)<t$ indicates that the

execution request of the VM v_i is submitted to a Cloud data centre by a user before the t^{th} time window, at(vi) + rt(vi) > t indicates that the VM v_i has not completed execution at the beginning of the t^{th} time window. Use ω_{it}, γ_{it} and λ_{it} respectively represents CPU processing power demand, memory size demand and bandwidth demand in the t^{th} time window. In the Cloud Computing environment, since the resource requirements of the virtual machine dynamically change, so the value of ω_{it}, γ_{it} and λ_{it} are change either in different time windows. Data from Google Cluster Job Load Data Set [134, 135] was used, which collects the data of CPU and memory during the fixed time interval.

Let $V_{j,t}$ represent the set of VMs running on compute node cn_j at the beginning of the t^{th} time window, then for any t (1r job load data sets. r>$_{j,t}$ = $V_{j,t-1} + V_{j,in} - V_{j,out}$, where $V_{j,t-1}$ represents the set of virtual machines running on cn_j at the beginning of (t - 1) time windows, and $V_{j,in}$ represents the set of VMs running on cn_j that are newly placed at the beginning of the time window. $V_{j,\,out}$ denotes the set of VMs that have been migrated out of the compute node cn_j or have finished execution before the time window starts. The initial placement of virtual machine requests is done at the beginning of each time window.

Definition 4-4: Thermal Computing Nodes, (TCN). If the sum of the resources required by all the VMs running on a computing node is greater than the capacity of the computing node, the computing node is called a thermal computing node. Once a compute node becomes a thermal compute node, SLA violation occurs. In order to avoid increasing the SLA due to the thermal computing node, it is necessary to migrate some VMs running on thermal computing nodes to other computing nodes with sufficient resource capacity through online migration technology.

Definition 4-5: VM Placement Policy. The VM placement policy refers to a mapping relationship between the virtual machine set V(t) and the computing node set CN(t) within the t^{th} time window, which can be indicated with the matrix $X(t)=(x_{ij})_{n(t)*m(t),}$ where m(t) and n(t) respectively represents the number of compute nodes used by the Cloud data centre and the number of VM running in the Cloud data centre during the t^{th} time window. If the virtual machine v_i is placed on the compute node cn_j, then $X_{ij}=1$; otherwise, $X_{ij}=0$.

Definition 4-6: VM Integration. Virtual machine consolidation means migrating VMs from the compute nodes which have a smaller number of active VMs through online migration technologies, and shutting down idle computing nodes so as to achieve the goal of reducing energy consumption of Cloud data centres. The formalization of VM consolidation can be descripted as follows: In the t^{th} time window, the set of VMs running in the Cloud data centre is V(t) and n(t)=|V(t)|, and the set of computing nodes used in Cloud data centre is CN(t), and m(t)=|CN(t)|, the current virtual machine placement strategy is $X(t)=(x_{ij})n(t)*m(t)$. The virtual machine consolidation is to find a solution under the constraints of multiple constraints, and get a new placement strategy X'(t) from the set of virtual machines to optimize the objective function.

Definition 4-7: States of CNs (Computing Nodes). The computing nodes in the Cloud data centre can be in a normal power consumption state or a low power consumption state. For any computing node $cn_j \in CN$, if $V_{j,t} \neq \Phi$ t that is, the computing node cn_j is running VMs during the t^{th} time window, then cn_j is a computing node that is on working which means it is in normal power consumption state; Otherwise, if $V_{j,\ t}=\Phi$, the computer node cn_j is not used within the t^{th} time window, so it is in a low power state such as shutdown or sleep. The vector $S(t)=(s_1(t), s_2(t)..., s_M(t))$ is used to calculate the node state in the t^{th} time window, which is an M-dimensional vector $s_j(t)$ (1e window, whiccomputing node cn_j is in the normal power consumption state in the t^{th} time window, then $s_j(t)=1$; otherwise, $s_j(t)=0$. The computing node state vector S(t) can be calculated by a given virtual machine placement strategy $X(t)=(x_{ij})n(t)\times m(t)$, where $s_j(t)$ is calculated as the formula (4-3):

$$Sj(t) = \begin{cases} 1, if\ \sum_{i=1}^{n(t)} xij \geq 1; \\ 0, other. \end{cases} \quad (4\text{-}3)$$

The migration necessity VM consolidation problem studied in this chapter refers to how the Cloud data centre operating environment changes dynamically during the entire operation cycle (T). This includes the arrival of new VM requests, the execution of VMs, the dynamic changes of resource requirements during the VM execution process, as well as the completion of the VM request to determine the condition of VM integration, and to find a suitable VM placement strategy X (t), based on meeting a variety of restrictions and avoiding the unnecessary VM migration costs, keep the number of computing nodes in t^{th} time as low as possible.

4.2.2 Objective Functions and Constraints

Virtual machine consolidation generally aims at reducing the energy consumption of Cloud data centres. The optimization goal is usually to minimize the number of active computing nodes [112, 126, 127, 132, 136]. Consistent with most existing VM consolidation studies, this chapter also uses the minimizing number of active compute nodes as the optimization goal for VM integration. It should be pointed out that most of the existing research focuses on the consolidation of static virtual machines, that is, executing single VM consolidation in the current state of Cloud data centres. What this chapter studies is to adjust the Cloud data centre's running status through multiple VM integrations within the entire operating cycle T (T= {1, 2, 3,..., |T|}) of the Cloud data centre. Therefore, the proposed VM consolidation model of migration necessity is to minimize the number of active compute nodes in each time window, i.e. within each time window $t \in T$, $\min \sum^{M}_{j=1} s_j(t)$.

Within each time window t (1indow, iating cycVM placement strategy X'(t) obtained by virtual machine consolidation must satisfy the following constraints:

(1) Ranges: For any i,(1Forn(t)) and any j,(1t)) and an$x'_{ij} \in \{0,1\}$, sj(t)$\in\{0,1\}$, that is,the value of x'_{ij} and $s_j(t)$ can only be 0 or 1.

(2) The VM must be relocated: for any i, (1for any ly$\sum_{j=1}^{M(t)} X_{ij}=1$, $i \neq j$) which means that the VM v_i must be deployed to another compute node.

(3) The capability limitation of computing node: For any computing node $cn_j \in CN(t)$, cn_j must satisfy $\sum_{j=1}^{n(t)} X_{ij} * \omega(t) \leqslant \Omega_j$, $\sum_{j=1}^{n(t)} X_{ij} * \gamma(t) \leqslant \Gamma_j$, and $\sum_{j=1}^{n(t)} X_{ij} * \lambda(t))\Lambda_j$, that is, the sum of the CPU, memory and bandwidth resource requirements of all VMs on one compute node cannot exceed its CPU and memory processing capabilities.

(4) SLA Violation Percentage (SVP). When the computing node's capacity is lower than the resource demand of VMs running on it, it will destroy the service level agreement (SLA) between the user and the Cloud provider. In this case, some virtual machines need to be migrated to other compute nodes with sufficient resource capacity through the online migration technology, so as stop increasing the SLA. $SLA_j(t)$ is used to indicate whether the SLA occurs in the t^{th} time window of the computing node cn_j, and the equation is shown in formula (4-4).

$$SLAj(t) = \begin{cases} 1, if\ \sum_{i=1}^{n(t)} xij * \omega(t) > \Omega t) > \text{or} \sum_{i=1}^{n(t)} xij * \gamma(t) > \Gamma j,\ \text{or} \sum_{i=1}^{n(t)} xij * \lambda(t) > \Lambda j\ ; \\ 0, other. \end{cases}$$

（4-4）

Therefore, in order to meet the user's requirement, the SLA of computing node should be reduced as much as possible during the virtual machine integration. In the algorithm proposed in this chapter, the SLA is restricted in such a way that SLA of the same computing node cannot exceed the time length of single time window.

(5) Migration consumption (MC). The current online migration of VMs is implemented based on the memory Pre-Copy mechanism. VM migration will increase the communication burden on the Cloud data centre and bring additional migration energy consumption. In addition, during virtual machine migration, the VM's performance will also be impact, even a short downtime. However, frequent Cloud virtual machine migration may occur during the daily operation and maintenance of Cloud data centres. In order to reduce its impact, the cost of migrating VMs should be reduced as much as possible during the VM integration. Moreover, this chapter also

take into account the factor of VM's remaining execution time, that is, in the consolidation process, the VM migration is unnecessary if its remaining execution time is not short than a time window.

4.3 VM Migration Necessity-based Dynamic Scheduling Algorithm

4.3.1 Algorithm Description

According to the description of VM consolidation problem in the previous section, this chapter aims at solving the dynamic process, that mainly includes the initial deployment when a new virtual machine request arrives, the dynamic changes in resource requirements during to the VM execution process, and the VM consolidation and thermal computing node elimination caused by the end of execution. Based on the above process of VM consolidation problem, this chapter proposes a multi-stage consolidation algorithm called VM migration necessity-based dynamic scheduling algorithm (VMMNDSA). The algorithm is a multi-stage algorithm that includes four stages: pre-processing, thermal compute node elimination, initial virtual machine placement, and virtual machine integration. The following describe the various stages of the VMMNDSA algorithm.

(i) Pre-Processing

In the Cloud Computing environment, the VM resources demand dynamically changes with time. The Google cluster load data set adopted in this chapter also reflects the dynamic changes in the resource requirements of the VMs during its operation. The Google Cluster workload dataset was published by Google after VM log files processing on its Cloud Computing cluster system. Therefore, in the pre-processing stage of the VMMNDSA algorithm, the entire operating cycle T of the Cloud data centre is also divided into |T| time windows according to the unit time length $t\Delta$ by "Definition 4-1" in Section 4.2.1. In this way, although the demand for CPU: ω_{it}, memory: γ_{it} and bandwidth: λ_{it} of VM v_i is different in different time windows, it can be regarded as fixed during the t∆ time of the time window.

The Pre-Processing of the VMMNDSA algorithm mainly accomplishes two tasks: First, judging if the VM running in the Cloud data centre is over. If the execution of the VM ends, then release the computing node. Second, according to the real-time data in this window, dynamically adjust the CPU and memory resources occupied by the running VM. The Pre-Processing algorithm is executed at the beginning of each time window.

The pseudo-code description of the algorithm is shown in Algorithm 4.1. The input is the current time window t (1 th|T|) and the set of VMs, V (t-1), running in the Cloud data centre during the (t−1)th time window. The output is a set of virtual machines V(t) running in the Cloud data centre at the beginning of the tth time window. The specific execution process of the Pre-Processing algorithm is as follows: at the beginning of each time window t, first free all computing node resources occupied by the executed virtual machine within the (t-1)th time window, as shown in step 2-7 of Algorithm 4.1, where the judgment condition in step 3, at(vi)+rt(vi)>(t-1), refers to that the VM v_i is still executing at the beginning of the (t-1)th time window, and at(vi)+rt(vi)<t refers to the virtual machine v_i at the beginning of the tth time window, the execution has finished. Second, the Pre-Processing algorithm will adjust the CPU and memory resources occupied by the current running VMs in the Cloud data centre at the beginning of each time window. This is shown as step 8-12 in algorithm 4.1.

Algorithm 4.1: Pre-Processing Algorithm

```
Input: the current time window, t(1npu|T|), V(t-1)
Output: V(t)
1.Intilizaing: V(t)=V(t-1);
2. Foreach VM vi∈V(t) do
3.   If at(vi)+rt(vi)>(t-1) && at(vi)+rt(vi)<t then
4.        Release all rescource occupied by vi
```

```
5. End foreach
6. Foreach VM vi∈V(t) do
7.    Adjust the CPU, memory and bandwidth that VM vi occupied
8. End foreach
9. Return V(t);
15. end if
```

(ii) Thermal Computing Nodes Removing

After the pre-processing operation with the VMMNDSA algorithm, due to changes in the resource requirements of the VMs running in the Cloud data centre, some computing nodes' capabilities may be insufficient and become thermal computing nodes. At this point, it is necessary to migrate the VMs running on the thermal computing node to other compute nodes with sufficient resource capacity to eliminate the thermal compute node. This process is called TCNR and the algorithm used is the Thermal Computing Nodes Removing (TCNR) algorithm.

The main factors that TCNR algorithm consider are the VM migration consumption and the VM remaining execution time. From Section 4.2.1, we know that the VM migration consumption consists of four parts: migration time (in seconds), downtime (in milliseconds), migration energy consumption (Unit: Joule) and performance loss of migration (unit less). Among them, the part of the performance loss of migration can be directly reflected by extending the execution time of the migrated VMs, because during the entire migration process, the performance loss of migrating a virtual machine is equivalent to a 10% reduction in the CPU usage of the VM. That is, the execution time of the virtual machine increases by one-tenth of the total time of the migration [5, 121]. For the other three parts, they differ in terms of unit and magnitude. In order to find the VM with the least migration consumption from the computing node, the concept of "Cost Factor (CF)" is introduced and defined as follows:

Definition 4-8: Cost Factor (CF). The cost factor of the VM refers to the value of VM migration comprehensive cost, including its CPU, memory and bandwidth requirements, as well as the migration time, downtime and migration energy consumption caused by the VM migration. If the computing node cn_j violates the SLA in the t^{th} time window, and the VM set running on the compute node cn_j is $V_{j,t}$, and $v_i \in V_{j,t}$, $CF_{mig}(v_i)$ represents the cost factor of the migration VM v_i. Its equation is shown in formula (4-5)

$$CF_{mig}(vi) = \alpha * (\omega_{it} + \lambda_{it})/(\gamma_{it}) + \beta * (T_{mig}(v_i) + T_{down}(v_i) + E_{mig}(v_i)); \qquad (4\text{-}5)$$

where ω_{it}, γ_{it}, λ_{it} respectively are the CPU requirement, memory requirement and bandwidth requirement of the VM v_i on the overloaded physical compute node cn_j in the t^{th} time window. In the formula, the greater ω_{it} and the λ_{it} are, the more the computing resource is consumed, and the migration can better relieve the computational resource load of the physical node. The smaller λ_{it} is, the smaller the migration overhead is due to less data need to be deal with. $T_{mig}(v_i) + T_{down}(v_i) + E_{mig}(v_i)$ are the part of migration cost which cannot be ignored, while α, β are the weighting factors of migration time and migration energy consumption ($\alpha+\beta=1$), which affect the migration cost factors.

The pseudo-code description of the TCNR algorithm is shown in Algorithm 4.2. The input of this algorithm is the current time window t(1The inp, the input is VM set V(t), current used computing node set CN(t), and the current VM placement strategy X(t). The output is a new VM placement strategy X'(t) after the thermal compute node is eliminated. The specific implementation process of the TCNR algorithm is as follows:

It begin with an initialization process, such that the value of each element in the new VM placement policy X'(t) is set as the value of the corresponding element of the current VM placement policy X(t).

Secondly, according to formula (4-6), it judges whether the current computing node in the Cloud data centre is a thermal computing node or not (Steps 2 and 3 of Algorithm 4.2). If the computing node cn_j is a thermal computing node, the following operations need to be performed on the computing node (Step 4-18 of Algorithm 4.2): finding the VM with the shortest remaining execution time from the VM set $V_{j,\ t}$ running on the computing node cn_j. and define it as v_l, its remaining execution time as T_{min}. At the same time, find all VMs who satisfy resource requirements from set $V_{j,t}$, calculate its cost factor, and mark the virtual machine with the smallest cost factor as v_k (Step 4-9 of Algorithm 4.2); if the VM v_l with the shortest remaining execution time can be executed within the tth time window, which means the remaining execution time T_{min} is less than tΔ or the migration time of the VM v_k, the SLA of the computing node cnj is within the limited range, and no migration is needed. Then the algorithm go to Step 2 to continue execution (Steps 10 and 11 of Algorithm 4.2); otherwise, the virtual machine v_k with the smallest cost factor is the VM to be migrated and the execute the CNSelection function to select a suitable destination compute node for the virtual machine v_k, which is denoted as cn_d ($cn_d \in CN$).

The virtual machine v_k is migrated from the source compute node cn_j to the destination compute node cn_d, and change the value of the corresponding position in the k-th row of the placement strategy X' (t), that is, let $x'_{kj}=0$ and $x'_{kd}=1$ (Steps 12-17 of Algorithm 4.2). The virtual machine v_k is migrated from the source compute node cn_j to the destination compute node cn_d, and change the value of the corresponding position in the k-th row of the placement strategy X' (t), that is, let $x'_{kj}=0$ and $x'_{kd}=1$ (Steps 12-17 of Algorithm 4.2). The virtual machine v_k is migrated from the source compute node cn_j to the destination compute node cn_d, and change the value of the corresponding position in the k-th row of the placement strategy X' (t), that is, let $x'_{kj}=0$ and $x'_{kd}=1$ (Steps 12-17 of Algorithm 4.2).

$$\sum_{i=1}^{n(t)} xij * \omega(t) > \mathrm{ij} * \mathrm{r} \sum_{i=1}^{n(t)} xij * \gamma(t) > \Gamma\,\mathrm{j}, \mathrm{or} \sum_{i=1}^{n(t)} xij * \lambda(t) > \Lambda\,\mathrm{j}\ ; \quad (4\text{-}6)$$

Algorithm 4.2: Thermal Computing Node Removing (TCNR) Algorithm

```
Input: Current time window t(1(pu|T|), V(t), CN(t), X(t)
1. Initialization: X '(t) = X(t);
2. foreach compute node $cn_j \in CN(t)$ do
3.   if, according to formula (4-6), the compute node $cn_j$ is the thermal computing node then
4.      foreach virtual machine $v_i \in V_{j,t}$ do
5.       Calculate the VMs' remaining execution time to get $v_l$ with the shortedt
         execution time and record as $T_{min}$.;
6.         if the migration of VM $v_i$ can remove the SLA of computing node $cn_j$ then
7.            Calculate $CF_{mig}(v_i)$ according to formula (4-5), and record the virtual
              machine with the s mallest $CF_{mig}(v_i)$ value as $v_k$;
8.         end if
9.      end foreach
10.     if Tmin$\leqslant$ t$\Delta$ or Tmin$\leqslant$ Tmig($v_k$) then
11.        continue
12.     else
13.        $v_k$ is the VM to be migrated.
14.        $cn_d \leftarrow$ CNSelection($v_k$);
15.        Move the $v_k$ from the source compute node $cn_j$ to the destination compute
            node $cn_d$.
16.         updata the value of the corresponding position in the k-th row of the
            placement strategy X' (t), that is, let $x'_{kj}$=0 and $x'_{kd}$=1;
17.     end if
18.   end if
19. end foreach
```

In step 14 of the TCNR algorithm, the CNSelection() selection function is used. This function aims at maximizing the computing node's resource utilization based on the Best Fit Descending (BFD) method for each VM to be deployed. In this way, load balancing between these computing nodes can be ensured while reducing the number of computing nodes used. The pseudo code description of the CNSelection() function is shown in Algorithm 4.3. The input is the virtual machine v_k who need to be placed, and the set of compute nodes CN(t) used by the current time window t. The output is the compute node cn_j where v_k is going to be placed. The specific implementation process is as follows:

(iii) Target computing node selection

When considering a federated Cloud environment, the virtual machines allocated to the users' tasks can be migrated either to physical resources of the users' current Cloud provider or to physical resources of different Cloud providers. Such an allocation of virtual resources to physical resources can lead to a cooperative model when users' virtual machines can be migrated anywhere or a competitive model when users' virtual machines can only be migrated to their providers' physical machines as the target computing node, which is expressed by Figure 4-4.

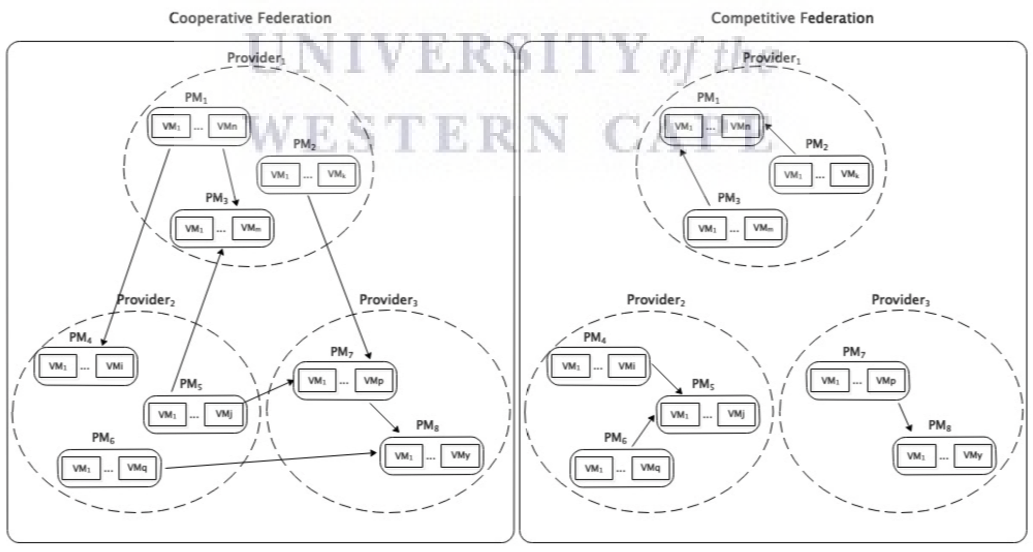


Figure 4-4 the structure of Cooperative and Competitive federation

Firstly, according to the federation type, we confirm the range of target computing nodes. Then we select from the computing nodes used by the Cloud data centre in the current time window t. The selection method is: find the compute node with the largest CPU, memory and bandwidth resource capabilities from the CN(t), and determine the remaining resources of the compute node. If its capacity can reach the resource requirements of v_k, while meeting the federation condition，then denote it as cn_j, return cn_j (Steps 1-6 of Algorithm 4.3).

$$cnj \in \{CN\text{-}CN(t)\}, \text{ and } (\Omega_j + \Gamma_j + \Lambda_j) = \max_{cnj \in \{CN\text{-}CN(t)\}} \{ \Omega_j + \Gamma_j + \Lambda_j \} \qquad (4\text{-}7).$$

Algorithm 4.3: Compute node selection function: CNSelection(v_i)

```
Input: virtual machine $v_k$, current time window t compute node set used CN(t)
Output: compute node where virtual machine $v_k$ is placed $cn_j$
1. For all compute nodes in CN(t), sort the CPU, memory and bandwidth resource
   capacities in descending order based on the formula (4-7). Assume that the order of
   the calculated compute nodes is $cn_1, cn_2, ..., cn_{|CN(t)|}$;
2. if the federation is cooperative
3.    foreach compute node $cn_j \in \{cn_1, cn_2, ..., cn_{|CN(t)|}\}$ do
4.        if CN $cn_j$'s remaining resource capacity can meet the resource requirements of
          virtual machine $v_k$
5.     return $cn_j$;
6. else if the federation is competitive
7.    foreach compute node $cn_j \in \{cn_1, cn_2, ..., cn_{|CN(t)|}\}$ do
8.        If CN $cn_j$'s remaining resource capacity can meet the resource requirements of
          virtual machine $v_k$, and $cn_j$ and $v_k$ belong to same PM provider
9.     return $cn_j$;
14. end if
15. return $cn_j$;
```

(iv) Virtual Machine Placement (VM Placement)

Algorithm 4.4: VM Placement Algorithm

```
Input: Current time window t, $V_{in}$(t-1), CN(t), current VM placement strategy X(t)
Output: New VM placement strategy X'(t)
1. Initialization: X'(t) = X(t)
2. For all compute nodes in CN(t), sort the CPU, memory and bandwidth resource
   capacities in descending order. Assume that the order of the calculated compute nodes
   is $cn_1$, $cn_2$, ..., cn|CN(t)|
3.  foreach VM $v_i \in V_{in}$ (t-1) do
4.      foreach compute node $cn_j \in \{cn_1, cn_2, ..., cn_{|CN(t)|}\}$ do
5.          If CN $cn_j$'s remaining resource capacity can meet the resource requirements
            of virtual machine $v_i$ then
6.              Place $v_i$ to $cn_j$
7.              $V(t) \leftarrow V(t) \cup \{v_i\}$, $n(t)= |V(t)|$;
8.              update the corresponding value of the X'(t) according to the placement of
                the VM $v_i$;
9.              break
10.         end if
11. end foreach
12.  If CN(t) can't find a suitable compute node to place VM $v_i$
13.         select the compute node with the largest sum of CPU, memory and
            bandwidth resource capabilities from the remaining unused compute nodes,
            denote it as $cn_p$, who satisfy formula (4-7);
14.         place $v_i$ to $cn_j$
15.         $V(t) \leftarrow V(t) \cup \{v_i\}$, $n(t)= |V(t)|$;
16.         add the computation node $cn_p$ to the set CN(t);
17.         $CN(t) \leftarrow CN(t) \cup \{cn_p\}$, $m(t)= |CN(T)|$;
18.         update the corresponding value of the X'(t) according to the placement of the
            VM $v_i$;
19.     end if
20. end foreach
```

```
21. return X'(t);
```

$V_{in}(t)$ represent the set of VM requests submitted by the user to the Cloud data centre within the tth time window, that is, for any VM $v_i \in V_{in}(t)$, satisfies t t is, for< (t +1). The task of the VM placement stage is to allocate the newly arrived VMs in (t-1)th time window to the appropriate computing node at the beginning of the tth time window. The VM Placement algorithm uses the Best Fit to Fall (BFD) algorithm to place the new arrived VM requests.

The pseudo-code description of the VM Placement algorithm is shown in Algorithm 4.4. The input is the current time window t, $V_{in}(t-1)$, the currently used set of compute nodes CN(t), and the current VM placement strategy X(t), outputs new VM placement strategy X'(t) after the newly arrived VM request set $V_{in}(t-1)$ is placed. The specific process of the VM Placement algorithm for is as follows: First, sort all compute nodes in CN(t) by their CPU, memory and bandwidth resource capacities in descending order. The order of the calculated compute nodes is cn_1, cn_2, ..., $cn_{|CN(t)|}$. Next, for each VM v_i in $V_{in}(t-1)$, judge from the front to the back of the computing node $cn_j \in \{cn_1, cn_2, ..., cn_{|CN(t)|}\}$. If cn_j's remaining resource capacity can meet the resource requirement of the virtual machine v_i, place v_i on the cn_j (algorithm 4.4, step 4-11); if a suitable computing node cannot be found in CN(t), place the v_i from the remaining unused compute nodes. Formula (4-7) selects the compute node with the largest sum of CPU, memory and bandwidth resource capabilities, denotes as cn_p, places v_i on the compute node cn_p, and adds the compute node cn_p to the set CN(t) (algorithm 4.4 Step 12-20). Finally, return the new VM placement policy X'(t) with all the VMs in the VM set $V_{in}(t-1)$.

The above is a detailed description of the various phases included in the VMMNDSA algorithm and the completion of each phase. The overall execution flow of the VMMNDS algorithm proposed in this chapter is as follows: At the beginning of each time window t, $(1 \leqslant t \leqslant |T|)$, first through the Pre-Processing algorithm to release the resources of the VM occupying the computing node that is executed in the previous time window (t-1), and simultaneously changes the size of the CPU, memory and bandwidth resources occupied by the executing VM. As a result of these changes, these compute nodes being used in the Cloud data centre may become thermal computing nodes. In this case, it is necessary to use the algorithm provided by the thermal compute node removing stage to remove these thermal compute nodes and avoid the SLA break. Then execute the initial deployment of the newly arrived VM requests in the $(t\text{-}1)^{th}$ time window with the VM placement algorithm. Finally, integrate VMs, close the idle computing nodes to achieve the purpose of reducing energy consumption.

4.3.2 Time Complexity Analysis

This section mainly analyzes the time complexity of the VMMNDSA algorithm. In order to analyze the complexity of the VMMNDS algorithm, we first let N = V(t), n = $V_{in}(t)$. From the description of the process of VMMNDS algorithm in describe in section 4.3.1, the algorithm includes a total of four separate stages: pre-processing, thermal compute node removing, initial virtual machine placement, and virtual machine integration. Firstly, the pre-processing stage will traverse all VMs in the virtual machine set V(t) running in the Cloud data centre, to judges whether the task is completed or not. Obviously, the time complexity of this stage is O(N). The second stage is to remove the thermal nodes, and it is necessary to traverse all the compute nodes and remove the thermal nodes. The time complexity is O (m·log N). Then, the time complexity of the initial VM placement phase is similar to that of the task scheduling algorithm analyzed in Section 4.4.2, which is O(n×m). In summary, the time complexity of the VMMNDS algorithm proposed in this chapter is $O(N + m\cdot\log N + n\cdot M)$.

4.4 Experiment and Result Analysis

We have discussed VM selection policy in the related work in 4.1.2, and VM allocation policy has been shown to be vital during the entire VM migration process.

In the experiment in this chapter, we choose two VM selection policies: MCM (Maximum Correlation Migration) and MMT (Minimum Migration Time), and two VM allocation policies: LR (Local Regression) and MAD (Median Absolute Deviation). We combined this two VM selection policies and two VM allocation policies into 4 VM migration algorithms, which are respectively LrMcm, LrMmt, MadMcm, MadMmt [5].

In order to evaluate the performance of proposed VMMNDSA algorithm, this section will test the VMMNDSA algorithm through simulation experiments, and compare the experimental results with the above four algorithms and the DVFS (Dynamic Voltage and Frequency Scaling) algorithm [5].

4.4.1 Experiment Parameter Setting and Performance

A. Experimental parameter settings

The simulation environment remains similar to those in the previous chapters. The compute node data and VM data used in this chapter's experiment are anonymized data from the Google cluster-loaded data set used in similar experiments [126, 128, 137]. A data centre with 30 PMs was set up and three different VM types as described on Table 4-2.

B. Table 4-2： experiment parameter of VM migration

Virtual Machine Setting		VM Number	VM MIPS	VM Pes	VM RAM	VM Bandwidth	VM Size
VM Types	Type1	20	500	1	870	100,000	2,500
	Type2	20	1000	1	1740		
	Type3	20	2000	1	1740		
	Type4	20	2500	1	613		
Host Setting		Host	Host	Host	Host	Host	Host Size

		Number	MIPS	Pes	RAM	Bandwidth	
Provider 1	10 Type1 Hosts	25	1860	2	4096	100,000,000	1,000,000
Provider 2	10 Type2 Hosts	25	2660	2	4096		
Provider 3	10 Type3 Hosts	25	2980	2	4096		
Provider 4	10 Type4 Hosts	25	3220	2	4096		
the cost of using processing	3.0						
the cost of using memory	0.05						
the cost of using bandwidth	0.001						
Max/Min Load Threshold	80%/0%						
$\alpha = \beta = 0.5$							

C. Performance indicators

In order to verify the effectiveness of the VMMNDSA algorithm proposed in this chapter, the following were used as metrics to compare the experimental results, similar [29] [117] [126].

1) The number of VM migrations: VM migrations count during the simulation.
2) SLA violation rate: ratio of computing node that violates the SLA during the entire operating cycle T to the total nodes in the system; equation (4-9). In

this chapter, the data centre's entire operating cycle T is divided into |T| periods of the same size, so reducing the time of SLA is equivalent to reducing the number of SLA.

$$\text{SLA} = \frac{\sum_{t=1}^{T} \sum_{j=1}^{m(t)} SLAj(t)}{\sum_{t=1}^{T} m(t)} \quad (4\text{-}9)$$

3) The VM migration mean time (ms): average migration time of all VMs during the whole simulation process, in millisecond (ms).
4) The energy consumption: includes the sum of the costs of all the migrated VMs in each time window, the total costs of all the VMs migrated in the entire operation cycle T.

4.4.2 Analysis of Results

Figure 4-5 shows the number of VM migrations. Compared with the five existing algorithms, the MadMMT algorithms had the most frequent migration at 5256 in cooperative federation and 1909 in competitive federation. The VMMNDS performed significantly better at 590 times in cooperative federation and 233 times in competitive federation, which is almost one-tenth of MadMmt. Compared with the least migrated LrMcm algorithm, VMMNDS VM migration count was only about a quarter of LrMcm for both cooperative and competitive federation. The results of this experiment highlight the characteristics of our algorithm: reducing unnecessary VMs migration. This is because when the VMMNDS algorithm selects the VM to be migrated from the thermal compute node, the factors of the remaining execution time of the VM are taken into account (eg, Steps 10 and 11 of Algorithm 4.2, and Steps 5 and 6 of Algorithm 4.5). In essence, VMMNDS does not unnecessarily migrate VMs that have a short remaining execution time.

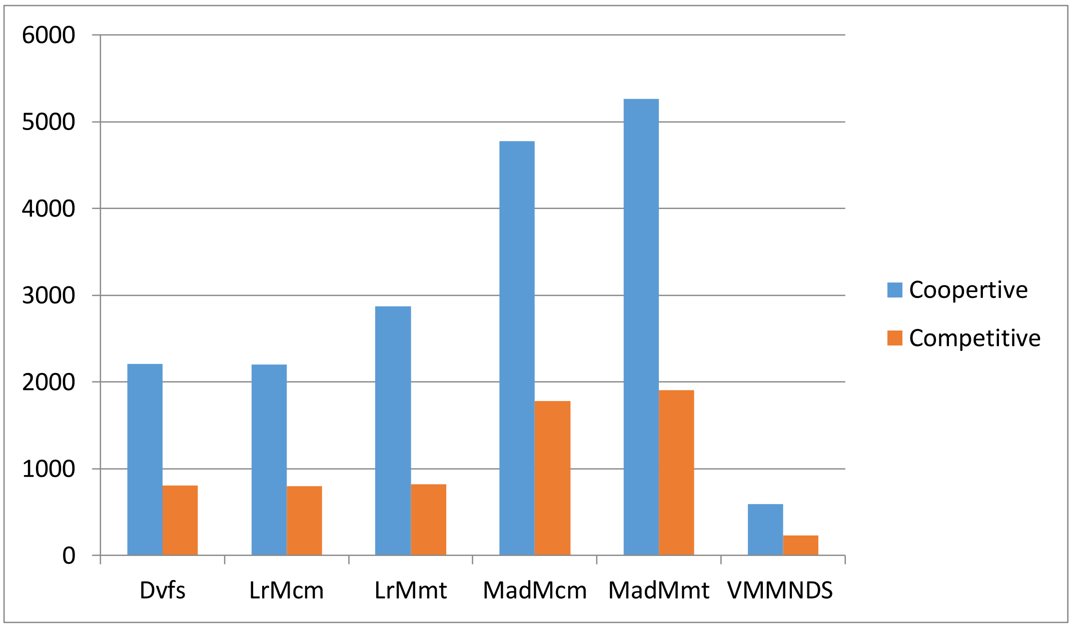


Fig 4-5. Migration times

Secondly, figure 4-6 shows that in terms of the SLA violation, five of the existing alogrithms: LrMcm, LrMmt, MadMcm, MadMmt performance better than VMMNDS in both cooperative and competitive federation. This is because when a compute node becomes a thermal compute node, the four algorithms will choose to migrate a certain number of VMs to avoid the SLA. This is done without considering the execution time remaining. VMMNDS on the other hand considers execution time. If a VM's remaining execution time is minimal, VMMNDS ignores the VM. This affects SLA and accounts for the slightly higher SLA violation rate than the five algorithms.

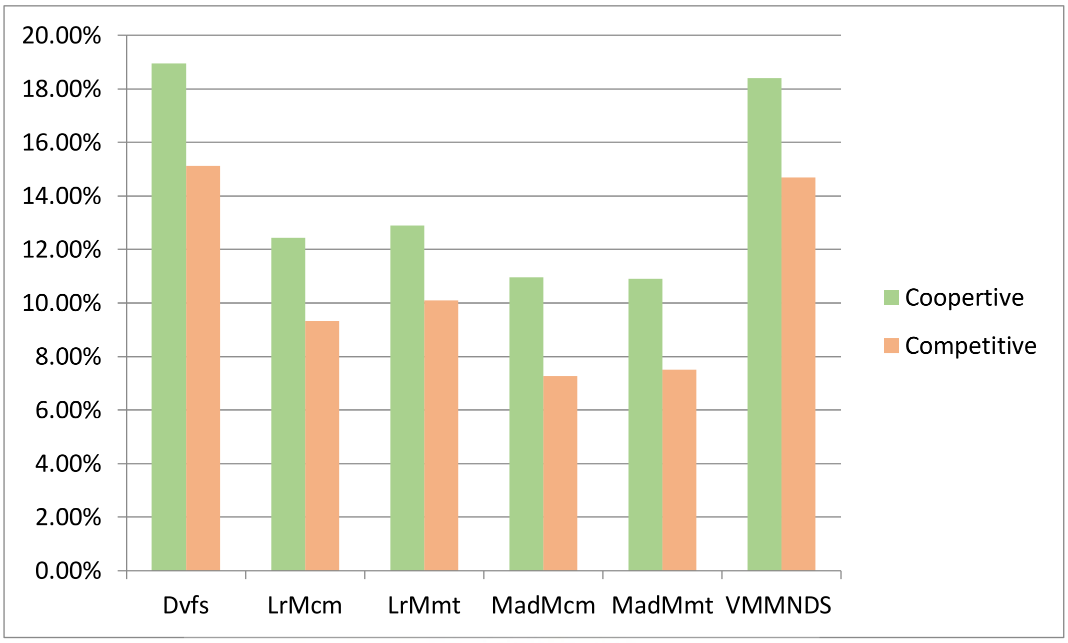


Fig 4-6 SLA violation rate

Figure 4-7 is the experiment result of the migration mean time of each VM. The MadMcm and MadMmt algorithms performance worst since they have the longest VM migration mean time in both cooperative and competitive federation. This is beacace they adopt the Median Absolute Deviation VM selection policy which is more complicated than others. VMMNDS on the other hand had the shortest VM migration mean time in both cooperative and competitive federation due to its efficient VM selecting policy and VM allocation policy.

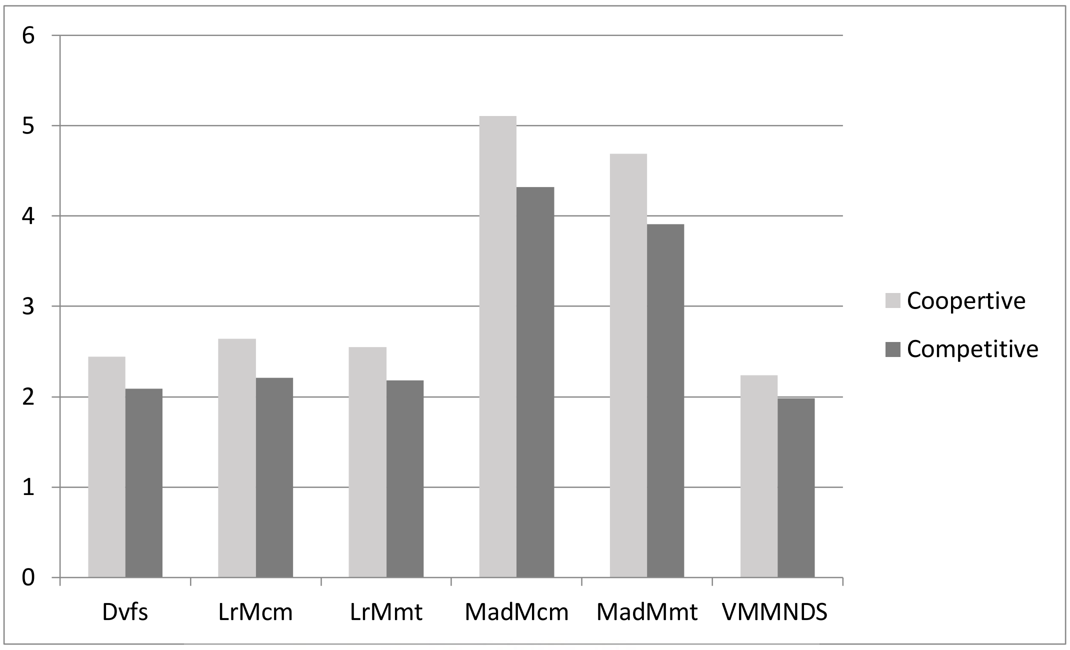


Fig. 4-7 VM migration mean time (ms)

From the results shown in Figure 4-8, the VMMNDS algorithm reduces the total migration cost (energy consumption) by 46.21% in cooperative federation and 44.85% in competitive federation, when compared to the MadMmt algorithm which has the highest energy consumption. Compared to the other four algorithms, our proposed VMMNDS on the average conserves 21.91% more energy under cooperative federation and 20.79% under the competitive federation. This is because the VMMNDS algorithm always chooses the VM with the lowest migration cost each time during the selecting VM migration process and avoid the unnecessary VM migration as much as possible, so the VMMNDS algorithm can obtain lower average migration cost. In addition, the VMMNDS algorithm in itself requires fewer migrations, which has been proven in Figure 4-5. It can be known that considering the impact of VM migration costs and remaining execution time in the selecting VM migration process can greatly reduce the additional consumption caused by unnecessary VM selections.

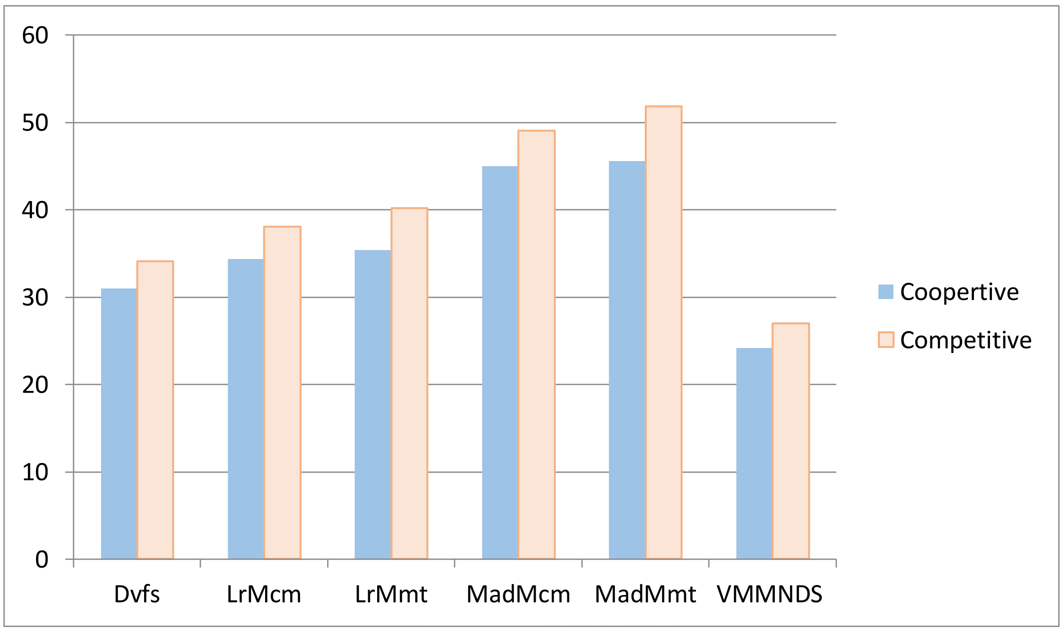


Fig. 4-8 Energy Consumption

4.5 Conclusion

Virtual machine consolidation is a commonly used method to effectively solve the high energy consumption problem of Cloud data centres. This chapter combined VM migration with VM consolidation and proposed the VMMNDMS model. This model considers both cooperative and competitive Cloud federations. Experimental simulations were used to verify the efficiency of the proposed model. Obtained results show that for both cooperative and competitive federations, our proposed VMMNDMS can greatly reduce the number of virtual machine migrations and migration mean time. This helps reduce the impact of migration on the overall performance of the Cloud Computing center and achieve more energy-efficient results, while keeping SLA at an acceptable level.

Chapter 5: IoT-based Fog Computing Model

5.1 Introduction：

With the maturity of wireless communication technology, Internet of Things has made continuous progress and breakthroughs. Terminal equipment is continuously been miniaturized, networked, and intelligently developed, hence able to support a wider range coverage and deployment scenarios. This terminal equipment, also known as sensors, collects data and uploads it on Cloud facilities, which have more sophisticated processors and sufficient memory resources. Over the years, the volume of data being transmitted has increased rapidly with the increase in the number of terminal devices. This has created problems of delay and congestion in a Cloud Computing environment [139]. Fog computing has emerged as a solution to these latency and distance related problems. By increasing the local computing and storage capabilities of the fog nodes and the edge devices can share some of the tasks previously solely handled at the Cloud data centre.

Stolfo used the idea of “fog” to resist hacking. He proposed the term “Fog Computing”, which was later used by Cisco to promote products and networks development strategy. The concept of “Fog Computing” was first proposed by Cisco in 2011 [34] and defined as a distributed computing infrastructure for the Internet of Things (IoT) that extends computing power and data analytics applications to the "edge" of the network. A Fog Computing model is equivalent to a local Cloud, where data management and done are controlled by local servers. Users can analyze and manage their data at any time, any place and in any way. The core idea of Fog Computing is “smart front-end”, which uses network devices or dedicated devices to provide computing, storage, and network communication services between Cloud servers and terminal devices, for making data storage and computing much closer to terminal acquisition, which greatly benefits reducing data transmission and storage overhead, improving application response speed, and saving network resources [140] [141].

Fog Computing is the middle layer between Cloud Computing and terminal layer of the IoT network, which is located at the edge of the network, close to the terminal. It is often combined with Cloud Computing to form a common network structure model as shown in Figure 5-1. The figure depicts the Cloud Computing layer, Fog Computing layer and terminal access layer Resource [142]. In the coverage area of the fog node, various intelligent terminals access the node and realize interconnection and intercommunication. In addition, Fog Computing layer is able to complete the direct computing processing to reduce the network transmission delay. The Cloud Computing layer, as the top-level supporting structure of Fog Computing, collects some statistical analysis data to the Cloud data centre, analyzes the big data globally, to coordinate the overall situation and implement resource allocation.

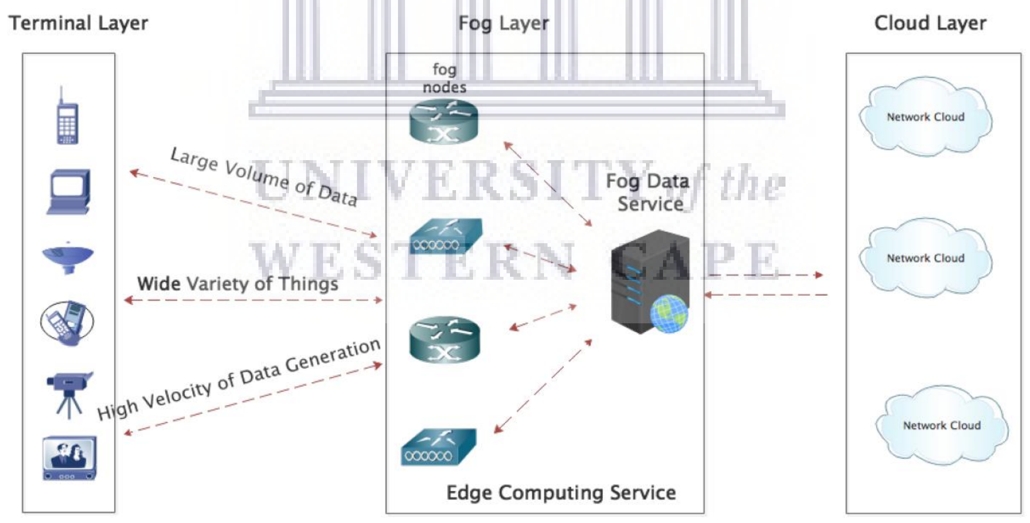


Fig. 5-1 the architecture of Fog Computing

There is a close relationship between Fog Computing and Cloud Computing. Many technologies in Cloud Computing can be directly applied in the fog. The Fog Computing node is actually equivalent to a small Cloud. Of course, Fog Computing also has its own unique characteristics and advantages [142] [143] [144] [145] [146] [147], including:

(1) Located at the edge of the network, hence can better solve the delay problem of real-time interaction of a large number of applications. This is especially vital for some applications that require high real-time performance, such as virtual simulation games, augmented reality, real-time monitoring and video conferencing.

(2) Extensive geographical distribution. This is a major different from Cloud and Fog computing. The Fog network structure is mainly distributed, with limited coverage, while the Cloud’s coverage area is often significantly larger. With the continuous intelligentization of terminal devices, various information collection devices are also developing toward mobile handheld devices, and the wide coverage is convenient for achieving more accurate positioning and geographic information perception.

(3) The number of massive nodes. Due to the wide geographical distribution, a large number of terminal devices and network nodes will constitute a large-scale Fog Computing network, and a large number of node accesses can also enhance the computing and storage capabilities of the Fog Computing network.

(4) Support for mobility. The development of terminal in form of smart devices and handheld is the future. The Fog Computing supports some necessary mobile communication technologies, including transmission protocols and identity authentication, to support communication between applications and mobile terminals.

(5) Heterogeneity. This enables communication between heterogeneous hardware and software devices in different forms and environments. It also facilitates efficient access of terminal devices and cross-domain convergence of application services to achieve interconnection and information sharing.

The terminal layer is often a wireless sensor network of the Internet of Things, with a multi-hop self-organizing network system composed of a large number of nodes deployed in the certain area and communicating wirelessly. The goal is to collaboratively perceive, collect, and preprocess information about the perceived objects in the network coverage area and send the information to the observer [148]. We assume that each node in the terminal layer has its own battery, and there are variations in power at specific time intervals due to different factors specifically the amount of activity performed by each node. The power level of a terminal node is dependent on how much work the node is doing. Factors that could overwork a node or increase its power depletion include: traffic, the number of child nodes connected to it, ambient temperature and humidity [149]. Energy consumption must always be considered as the main factor to design and measure IoT related protocols. With this in mind, this chapter proposes an IoT-based Fog Computing model, with the following specific contributions:

An IoT-based Fog Computing framework: a multi-layer framework for IoT-based Fog Computing environments that addresses issues related to: i) the topology of the terminal layer network and its impact on the routing of data in that layer; ii) the allocation of tasks uploaded from the terminal layer to resources (Fog nodes) in the Fog layer . The proposed framework is based on a model that minimizes the overall cost (delay, distance, energy) of completing the terminal tasks using Fog nodes (FN).

GA based Fog layer task scheduling strategy: A task scheduling strategy for the Fog layer using a modified Genetic Algorithm (GA) for matching tasks (uploaded from the sink nodes in the terminal layer) to corresponding FN is proposed. The task requests and the geographical location of the Fog and sink nodes are used as input for the modified GA, which outputs a binding scheme of tasks to resources (FNs). Implementation is done using CloudSim [63] and the relative efficiency of the new algorithm compared to the traditional Max-min algorithm and the Fog-oriented Max-min [150] algorithm is revealed.

A multi-sink LIBP Terminal layer protocol: This thesis proposes a novel collection tree protocol that builds upon LIBP [151] protocol to organize the terminal layer into a multi-sink IoT network. The objective is to improve the robustness and reliability of the terminal layer network and extend the battery life of the sink nodes. Simulations using Cooja [152] on the Contiki OS [153] are used to demonstrate the efficiency of the multi-sink protocol compared to the mono-sink LIBP protocol.

Our expectation is to improve the robustness of the underlying IoT networks' and safeguard it against nodes failures as well as extend the terminal nodes' life span. These are achieved through the use of multi-sink deployment, while reduction in processing delays and energy consumption are achieved by incorporating a Fog Computing layer.

5.2 Related work:

The Fog Computing was first proposed by Bonomi [34] in 2011. In order to solve the applicability of Platform-as-a-Service (PaaS), Hong *et al.* [154] proposed the concept of Mobile Fog, which realizes the connection of heterogeneous devices simplification, as well as on-demand dynamic expansion of applications, which enhances the ability to interconnect communications between heterogeneous devices and enhances the universal application of Fog Computing. Oueis [155] applied Fog Computing to the processing of load balancing to improve the quality of the user's network experience. Applications spanning Cloud and fog, such as Internet of Things (IoT) applications, are still provisioned manually nowadays, but Yangui *et al.* [156] proposed a Platform as-a-Service (PaaS) construct for hybrid Cloud/fog environment to automate applications.

As a result of the combination of IoT and heterogeneous devices, Abedin [157] addressed the utility based matching or pairing problem within the same domain of IoT nodes by using Irving's matching algorithm under the node specified preferences to endure a stable IoT node pairing. In terms of the communication distance, Intharawijitr *et al.* [158] defined a mathematical model of a Fog network and the important related parameters to clarify the computing delay and communication delay in Fog architecture. Deng [159] focus on the interplay and cooperation between the edge (fog) and the core (Cloud), they develop an approximate solution to decompose the primal problem into three subproblems to make a balance between power consumption and delay in a Cloud- Fog Computing system. Sarkar [160] and his group conducted theoretical modelling of the Fog Computing architecture and analyzed the delay and energy performance of the application in the Internet of Things. They have accumulated the experience in the design and wide application of the Fog Computing architecture. Due to the Cloud Computing's high degree of polymerization computing mode, it can not give full play to the resources of the edge device. Ningning *et al.* [161], therefore proposed a Fog Computing framework to turn physical nodes in different levels into virtual machine nodes. Their simulation demonstrated that dynamic load balancing mechanism can effectively configure system resources as well as reducing the consumption of node migration brought by system changes.

In terms of the terminal layer of the IoT, the ubiquitous sensing technology enabled by Wireless Sensor Network (WSN) technology is one of the indispensable parts [148]. Thanks to the recent adoption of various supporting wireless technologies, such as RFID tags and embedded sensors and actuator nodes, the Internet of Things has come out of its infancy and is the next revolutionary technology to transform the Internet into a fully integrated future Internet. In [162] a use case considering energy consumption measurements of RPL and CTP is presented and a metrics for several scenarios running both RPL and CTP was proposed. However, they did not consider the routing protocol’s robustness and reliability. Felici-Castell [163] and his team focused on analyzing different strategies to gather information from different topics. The trade-offs between the "always send" and "local buffer" methods are verified experimentally, which considering power consumption, lifetime, efficiency and reliability. Machado [164] proposed a routing protocol based on Routing Energy and

Link quality (REL). The end-to-end link quality estimation mechanism, residual energy and hop count are used to select routes to improve the reliability and energy efficiency of IoT applications. In addition, REL proposes an event-driven mechanism to provide load balancing and to avoid premature depletion of energy by nodes/networks. But they didn't take into account the affect of different number of sinks.

There are some new research work such as [165] [166] [167] [160] [168] which focus on the combination of Cloud Computing, Fog Computing and IoT. Yannuzzi [166] examines some of the most promising and challenging problems of IoT and explained the reason why current compute and storage models confined to data centres may not be able to meet the requirements of many applications. Their analysis is particularly centered on three interrelated requirements: mobility, reliable control and calability, then described why Fog Computing is necessary for IoT, and discussed the unavoidable interplay of the Fog and the Cloud in the future. In terms of the framework between IoT and Fog Computing, Donassolo [167] proposed an orchestration system which is called FITOR, which build a realistic fog environment while offering efficient orchestration mechanisms. Based on extensive experiments, they proposed O-FSP optimizes the placement of IoT applications and the related strategies in terms of provisioning cost, resource usage and acceptance rate. Sarkar [160] did some works on the Fog Computing architecture modelling, which is one of the new researches arean. They proposed a mathematical formulation for improving the balance between the latency and energy saving. And their experiment results proved that the proposed model can save around 40% energy consumption. However, they did not consider the energy consumption in the IoT layer. Aazam [168] proposed using Cloud of Things (CoT) to solve the IoT resource management problem, and introduced the architecture of a smart gateway with Fog Computing. Then they tested this concept based on upload latency, synchronization latency, jitter, bulk data upload latency, and bulk data synchronization latency. But they did not take into account the energy consumption around the fog nodes in Fog Computing layer.

5.3 Model Description:

In Cloud Computing, effective resource allocation is the main goal of achieving economic benefits, while the main features of Fog Computing are location awareness, mobility, low delay and distributed geographic location. Fog computing is not a replacement for Cloud Computing, but they reduce the disadvantages of Cloud Computing and make them more efficient. The model proposed in this chapter focuses on energy consumption from underlying sensors (which are in the terminal layer) to fog nodes (which are in the fog layer). So, the model is called IoT-based Fog Computing model (IoT-FCM). The Fog Computing layer aspect of the model is described in the subsequent sub-sections, while the terminal layer of the model is discussed in section 5.4.1. The Cloud layer was introduced in chapter 2 and 3.

5.3.1 Fog Computing layer model design

Figure 5-2 shows the architecture of the IoT-based Fog Computing model (IoT-FCM) proposed in this thesis. The processes performed in the figure are as follows: in the first step, the application tasks queue (generated by sink-nodes) will be sent to the Fog manager service which is in the Fog Computing layer. This service manager has information about all the Fog nodes. Using the modified GA algorithm (which will be introduced in next section), the Fog manager service generates the task scheduling simulation results. The FNs then executes the manager assigned tasks. Before introducing the specific process of task scheduling strategy of the modified GA for Fog Computing layer, some definitions and assumptions are first introduced.

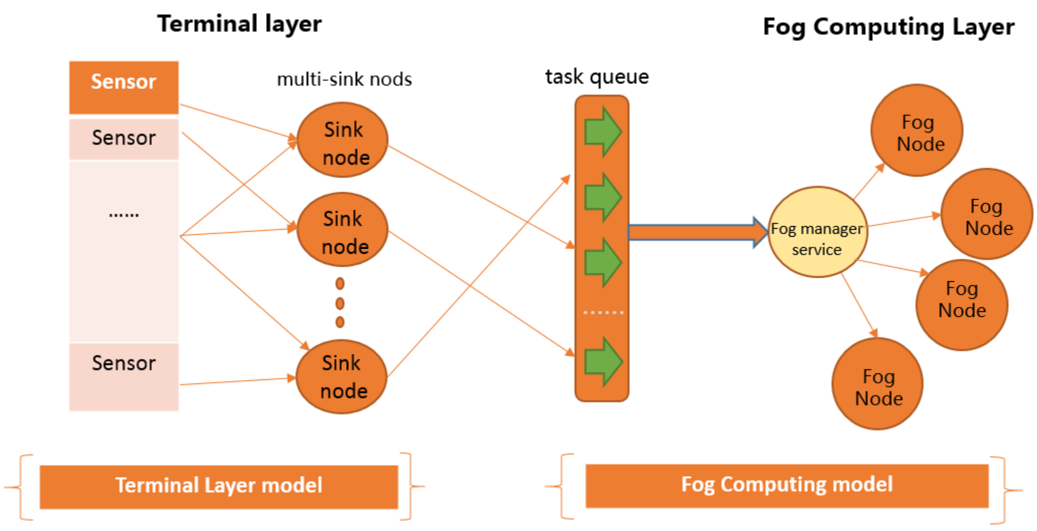


Fig. 5-2 The IoT-Fog architecture

5.3.2 Formalization of Fog Computing Layer Model

The Fog Computing layer model is constructed in this chapter, which focuses on three target parameter that decide it comprehensive performance. These parameters are: delay, energy and distance. Delay means the response time that users (sink nodes) have to wait after they submit their tasks. Energy is the total energy the target FN needs to finish its allocated tasks, while distance means the total distance of each user to their corresponding according to scheduling result. Suppose the Fog Computing system consists of the Fog Nodes, which can be represented as a set FN = {FN_1, FN_2, …, FN_N}, and the application tasks which are going to be scheduled can be represented as a set T = {t_1, t_2, …t_n}. The main factor affects delay in the execution time $ExeT_{ij}$, where i = 1, …, n and j = 1, …, N. $ExeE_{ij}$ is the energy consumption of t_i by FN_j.

The first quality factor considered is the total distance TD, which is the distance from users to their corresponding FN. This can be calculated by traversing all the tasks in set T. If (T_{iX}, T_{iY}) and Coord (FN_{jX}, FN_{jY}), denote the coordinates of user i and FN_j, respectively, then TD can be determined by using

$$TD = \sum_{i=1}^{n} \sqrt{((T_{iX} - FN_{jX})^2 + (T_{iY} - FN_{jY})^2)},\ TD<TDL \quad (5\text{-}1)$$

where T is the Task set, n is the number of tasks in set T, while connected to the j-th FN, and TDL is the total distance limitation.

From the Fog Computing characteristics, delay should be kept as low as possible. Tasks scheduling strategy therefore must aim at minimizing task completing time (execution time). FNs can hold more than one task at a time, the completing time is thus the execution time of such a task running on a FN whose execution time is the longest. The execution time ExeT of a task T by the FN can be described by using

$$ExeT = MAX\ \{\sum_{i=1}^{n} ExeT_{ij}; j \in N\},\ ExeT(T, FN) < DL, \quad (5\text{-}2)$$

where DL is the delay limitation, the summation of all the execution times ExeTij of the various tasks $T_i(_i \in n)$ running on a FN gives the completion time of each $FN_{j,\ j} \in$ N. The delay is obtained from the last FN_j to finish its tasks.

Energy controling is also a very important factor that needs to be considered while building Fog Computing models. Therefore, Fog Computing system should keep the energy consumption as low as possible. In addition, the scheduling energy consumption ExeE cannot be greater than the upper limit of electricity supply. The energy consumption for executing task set T by set FN is given by

$$ExeE = \sum_{i=1}^{n} \sum_{j=1}^{N} ExeEij, \quad ExeE(T, FN) < EL, \quad (5\text{-}3)$$

where $ExeE_{ij}$ is the energy consumed by FN_j, $j \in N$ to execute task T_i, $i \in n$ and ExeE is the energy consumed by all the FNs in executing their allocated tasks; DL is the energy limitation.

The three equations can be integrated into a fitness function which is defined by using

$$F(C)\begin{cases} = \alpha * \left(1 - \frac{\mathrm{ExeT(C)}}{\mathrm{DL}}\right) + \beta * \left(1 - \frac{\mathrm{ExeE(C)}}{\mathrm{EL}}\right) + \gamma * \left(1 - \frac{\mathrm{TD\ (C)}}{\mathrm{TDL}}\right), \\ \quad \text{if}, \mathrm{ExeT(C)} \le \mathrm{DL}, \mathrm{ExeE(C)} \le \mathrm{EL}, \mathrm{TD\ (C)} > \mathrm{TDL} \\ = 0, \quad \text{if } \mathrm{ExeT(C)} > \mathrm{DL}\ or\ \mathrm{ExeE(C)} > \mathrm{EL}\ or\ \mathrm{TD\ (C)} > \mathrm{TDL} \end{cases} \qquad (5-4)$$

where C is the vector of the special individual which includes one match between tasks and fog nodes (T, FN), F(C) is the fitness function means the fitness value of the vector C, which is used for measuring the score of the individual in the population, (1-ExeT(C)/DL) is the benefit of execution time which is considered as delay in the paper when finished the task scheduling, so the same (1-ExeE(C)/EL) is the benefit of energy and (1-TD(C)/TDL) is the benefit of distance. While α， β and γ are the weight factors to repectively adjust the importance of delay, energy consumption and distance.

5.3.3 The Modified Genetic Algorithm for Proposed Fog Computing Model:

As earlier mentioned, task scheduling in Cloud Computing environment is a NP hard problem. It is very difficult to find the best solution when the number of participants is large. The usual way is to apply various intelligent optimization algorithms to approach its optimal solution as the satisfactory solution. Genetic algorithm (GA) is one of these algorithms to get the approximate optimal solution. In this chapter, the classic GA is modified using a single fitness function, emanating from multiple fitness functions, as well as the generation of the third child of crossover in order to determine the optimal solution of the IoT-FCM model.
The modified genetic algorithm is presented as follows:

1. Initialization of the population.

Initialize the population and setting up of the relevant parameters, such as population size (P), probability of performing crossover (pc), probability of mutation (pm), as well as the evaluating fitness of every individual in the population are done. In GA, the proposed multi-target parameters correspond to multiple fitness. Hence, we use the equation 5-4 as the fitness function to evaluate each vector solution (chromosome).

2. Crossover

Following the principle of higher fitness is better, the second step involves choosing two individuals from the population as parents, upon which the crossover is executed to produce two children. In order to obtain an optimal solution, this chapter adds the third child to increase the diversity of the population which is generated by accumulating the parents corresponding gene values to generate a new child. The process is showed in Algorithm 1:

Algorithm 1: Using two n-dimensional arrays (C1, C2) to represent two randomly selected parent chromosomes

```
   input  : C1[n]= (c11, c21, ..., cn1), C2[n]= (c12, c22, ..., cn2)
   output : C (Third child)[n]
1  for $i=1; i \leqslant n; i{+}{+}$ do
2  |   C (Third child) [n]= (C1[n]+ C2[n])/2
3  end
4  return C (Third child) [n]
```

3. Mutation

There are many types of mutations such as Gaussian, Uniform mutation and Non-Uniform mutation [169]. In these mutations, the value of only a single gene in the chromosome is changed to improve its fitness. The effect of this on the entire chromosome is minimal especially with large population size or when the solution is close to stability [169]. We modified the mutation process, changing the single-gene mutation to multi-gene mutation. We then generate multi-mutated chromosomes to replace chromosomes with the lowest fitness value in the population. This reduces the impact on optimal values, while greatly expanding the search range and simultaneously reducing premature convergence to a local optimal solutions. The main purpose of mutation is to generate new genes when inheriting from parents. The mutation can be defined in equation 5 as follows:

$$C_{m1}(n)=(c_1,c_2,\ldots,c_n)+(x_1(\triangle c_1-c_1),x_2(\triangle c_2-c_2)),\ldots,x_n(\triangle c_n-c_n))) \,, \qquad (5\text{-}5)$$

where x1, x2,… xn ∈ {0,1}, and Δc1, Δc2,…, Δcn are the random numbers within the limits of gene in the chromosome. Then we can generate 4 different children by adjusting the number of x. The first mutated child has 1/4 of its genes (x) randomly set to 1, while the other genes are set zero. The second mutated child has 1/2 its genes randomly set to 1 and the others set to 0. The third mutated child has 3/4 randomly set to 1, while the fourth has all its genes set to 1.

Since we have 4 mutated children, we then select 4 chromosomes with the smallest fitness value from the population and compare with the fitness values of our 4 newly generated mutated children. After the comparison, we put 4 chromosomes with the highest relative fitness value back into the population to get a new population. The process is showed in Algorithm 2.

```
Algorithm 2: mutation
  input  : C[n]= (c, c, . . . , cn), population
  output: new population
  int Array Cm[8]
  Using equation (5) get the 4 chromosomes and put them in Cm[1], Cm[2], Cm[3], Cm[4]
  Traversing the population to get the 4 chromosomes with smallest fitness value, put them in
    Cm[5], Cm[6], Cm[7], Cm[8]
  for $i=0; i < 7; i++$ do
    for $j = 0; j < 7 - i; j++$ do
      swap Cm [j], Cm [j + 1]
    end
  end
  put the chromosomes in Cm[5], Cm[6], Cm[7], Cm[8] back to the population
  return the new population
```

4. Merging

In this phase, we merge the new chromosome population set generated by the crossover and mutation operations. Afterwards, the best chromosomal individuals who have the highest value of F(C) are select to be retained as population for the next generation.

Steps 2 to 4 are repeated till the simulation ends.

5.3.4 Terminal layer design of this model

This section will introduce the part of our proposed model design of the terminal layer. We use the modified LIBP [147] as the protocol for node communication in our model. The principle of LIBP is to make each node below the sink node select the parent node with the fewest child nodes in order to get the balance tree. The LIBP algorithm can beiefly be described as follows:

a. The spanning tree is rooted at the sink node by signing or marking its name in the beacon message, though recursively broadcasting and recording the parent node of the beacon message.
b. Select the parent with the fewest number of children to promote least traffic interference.

A routing in WSNs that solves a local optimization problem using a weight associated with a measure of interference (using number of children) is described in LIBP, but it does not consider other reliability constraints, such as energy efficiency, and nodes' robustness, nodes will need a lot of energy when they communicate with nodes at greater distances within their range and if sink node is offline, the system will stop working. The modified LIBP proposed in this chapter attempt to solve these limitations by adding multiple sink nodes to improve the system's battery life and improve the robustness of the entire system.

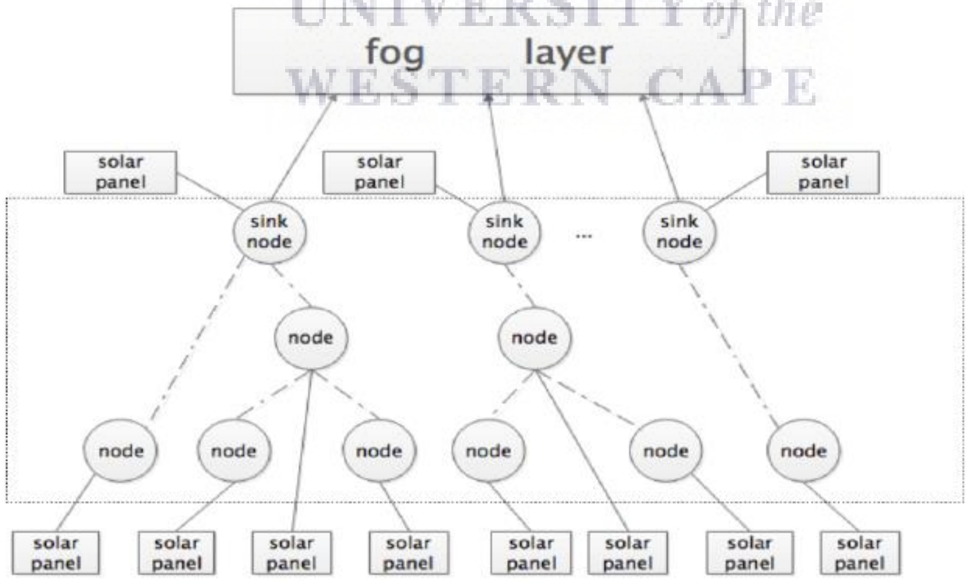


Fig. 5-3 Proposed terminal layer model.

Figure 5-3 presents the detailed structure of the terminal layer, with the following characteristics:

a. At least two nodes with GPRS: Terminal layer should have at least two nodes capable of transmitting IP packets to a fog node.
b. Only nodes with GPRS can become sink nodes. The sink node is selected based on whether the node has the lowest temperature and highest energy.
c. Each node should have a solar panel for energy regeneration.

5.4. Simulation result:

5.4.1 Simulation environment

CloudSim [63] was used as the simulation tool, while simulation hardware remained similar to those in the previous chapters. Performance configuration and computing power parameters were adapted from CloudSim and are shown on Table 1.

Table 5-1: Performance Configuration and Computing Power Parameters (adapted from CloudSim)

Fog Nodes:	$Node_1$	$Node_2$	$Node_3$	$Node_4$	$Node_5$	$Node_6$	$Node_7$	$Node_8$
Pes	2	4	2	4	2	2	2	2
Mips	550	300	650	350	750	800	850	900
Energy Cost	10	12	14	15	16	18	20	22
Coordinates	{10,10}	{10,40}	{10,70}	{40,10}	{70,10}	{70,40}	{70,70}	{60,80}

In the simulator, the application task parameters include task ID, task length and coordinates, in which task length use Millions of Instruction (MI) as a unit. Task length means the number of basic instructions of task scheduling requests. For this work, task lengths were set to 1000. We simulate the coordinates of FNs in an area, such as a city or a university, so we limit the range of FN in {0-100}, and randomly generate the coordinates of 100 FNs, which are shown in Table 5-2.

Table 5-2: Task Coordinates

Task number	Coordinates	Task number	Coordinates	Task number	Coordinates	Task number	Coordinates	Task number	Coordinates
0	{93,31}	20	{36,75}	40	{73,06}	60	{53,49}	80	{61,21}
1	{32,96}	21	{44,23}	41	{77,45}	61	{52,12}	81	{62,96}
2	{14,11}	22	{31,23}	42	{77,89}	62	{54,11}	82	{64,11}
3	{52,21}	23	{35,23}	43	{72,34}	63	{12,63}	83	{62,48}
4	{50,21}	24	{36,21}	44	{70,21}	66	{10,21}	84	{63,90}
5	{43,90}	25	{33,90}	45	{73,90}	65	{53,93}	85	{67,53}
6	{10,61}	26	{31,21}	46	{77,62}	66	{58,34}	86	{84,70}
7	{96,59}	27	{36,59}	47	{76,59}	67	{50,61}	87	{64,10}
8	{39,83}	28	{49,83}	48	{76,78}	68	{51,24}	88	{63,46}
9	{71,34}	29	{34,51}	49	{11,34}	69	{42,83}	89	{12,37}
10	{23,31}	30	{43,31}	50	{83,31}	70	{43,51}	90	{13,14}
11	{22,96}	31	{32,96}	51	{92,96}	71	{44,67}	91	{39,57}
12	{24,11}	32	{14,11}	52	{96,75}	72	{44,11}	92	{17,11}
13	{23,83}	33	{59,39}	53	{92,21}	73	{49,87}	93	{52,31}
14	{20,21}	34	{54,52}	54	{95,24}	74	{44,59}	94	{50,61}

15	{23,90}	35	{43,90}	55	{93,90}	75	{53,12}	95	{44,23}
16	{28,95}	36	{10,61}	56	{90,61}	76	{40,61}	96	{13,71}
17	{26,59}	37	{96,59}	57	{45,32}	77	{49,56}	97	{95,69}
18	{66,66}	38	{57,74}	58	{99,83}	78	{49,83}	98	{32,53}
19	{28,45}	39	{75,23}	59	{68,21}	79	{41,34}	99	{63,31}

GA operational parameters used are as follows: a population size of 100; mutation probability of 0.01; maximum iteration number of the algorithm was set to 1000; weighting factors set as: α=β=γ=1/3, delay limitation, energy limitation and distance were respectively set to 50, 2000 and 5000.

5.4.2 Simulation Results

In order to evaluate the proposed IoT-FCM model, simulations of both the Fog Computing layer and terminal layer were done. In this section, we show the simulation experiment results of Fog Computing layer using delay (makespan), sum of distance, sum of energy consumption as metrics. Then we proved the effectiveness of our proposed GA optimized IoT-FCM model by comparing it with traditional Max-Min algorithm and improved Fog-Oriented Max-Min algorithm. In task scheduling problem, the traditional Max-Min algorithm usually select the makespan as the main parameter to achieve the relative optimal solution. The Fog-Oriented Max-Min algorithm as used in this thesis considers multiple parameters (including delay, distance and energy consumption) to calculate the relative optimal solution.

Figure 5-4 shows the delay of the three different algorithms. Here delays as a result of signal transmission time were not considered because they are too short thus negligible; rather the focus was on task execution time at the FN. The proposed algorithm is aimed at minimizing the distance and energy consumption. In Figure 5-4, when compared to the two other algorithms (Fog-oriented Max-Min and Max-Min), IoT-FCM sacrified speed for better performance in distance and energy. It is shown to increase execution time by an average of 17.5% compared to the other algorithms. However, when a 100 tasks were submitted, it was at par with the other algorithms, as shown by the converged curves in Figure 5-4.

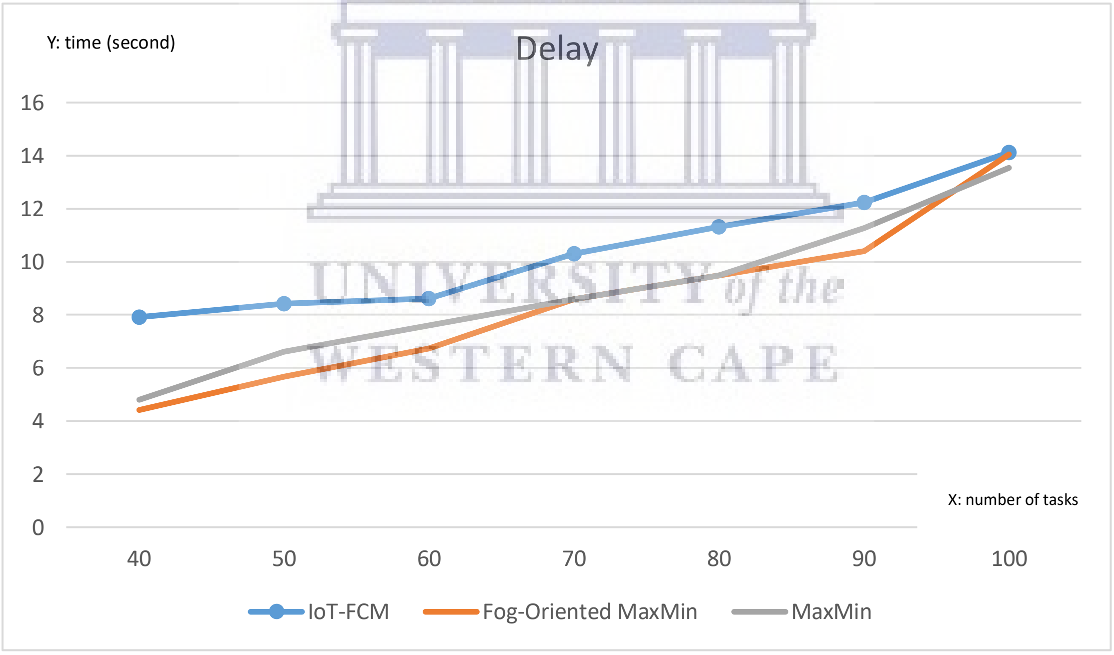


Fig. 5-4 Delay

Figure 5-5 shows the results for the total distance form users to their corresponding FNs. One of the benefits of Fog Computing relative to Cloud Computing is that it is closer to the terminal [29]. Hence minimizing the distance is vital. From the results, comparing IoT-FCM with the two other Max-Min algorithms, IoT-FCM is seen to have an advantage over the others for all submitted tasks as the lower the distance between terminals and corresponding FNs, the better the algorithm. Comparing the closeness of terminals to their FNs; using IoT-FCM, terminals are about 50% closer to their FNs when 40 tasks are submitted versus the two other algorithms and 38% closer versus Fog-Oriented Max-Min and 55% closer versus Max-Min when 100 or more tasks are submitted. The mildness of the curve also proved the stability and predictability of the IoT-FCM model.

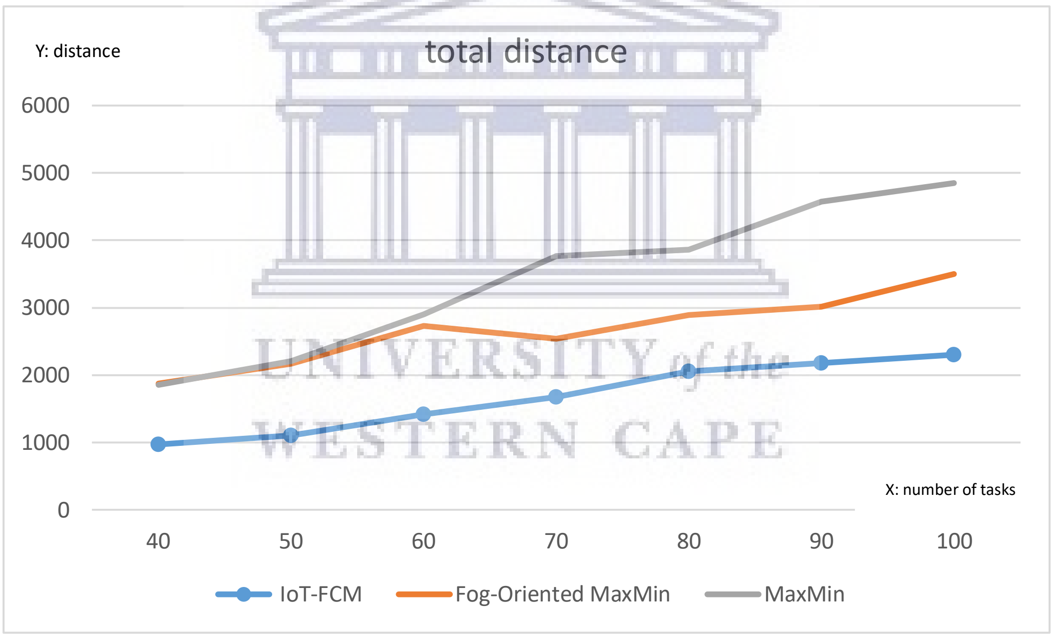


Fig. 5-5 The sum of the distances from each user to their corresponding fog node

Another important factor of Fog Computing considered in this work is the energy consumption which is shown in Figure 5-6. The energy consumption of all the Fog nodes in the system was taken into consideration. Obtained results show that the proposed algorithm has some advantages in terms of energy consumption during the whole test. On the average, IoT-FCM conserved about 100 mAh of energy compared to Fog-Oriented Max-Min for all submitted tasks and about 200 mAh when compare to the Max-Min algorithm.

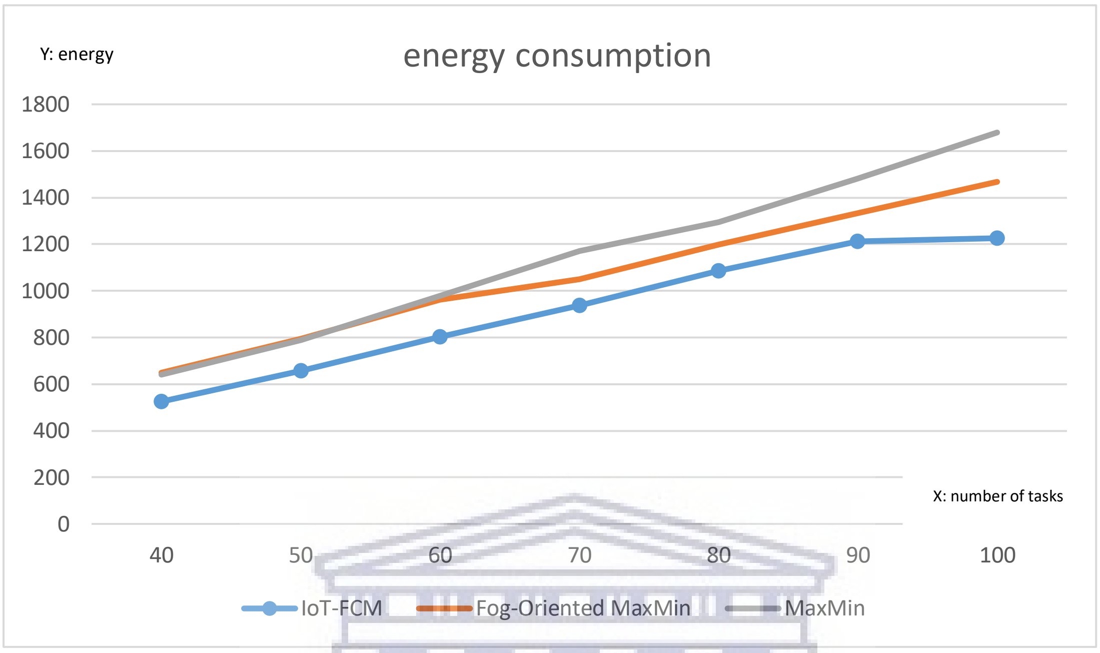


Fig. 5-6 the sum of the energy consumption of all fog nodes

5.4.3 Experimental Findings of Multi-Sink nodes.

As earlier stated, experimental simulation of the terminal layer was carried out using the Cooja on Contiki [153], which is a simulator embedded in Ubuntu 16.04 operating system. Using Cooja [152] we were able to implement and test the robustness and energy efficiency of the Multi-sink LIBP used at the terminal layer. Two categories tests were carried out: robustness and energy efficiency. The results are as follows:

Robustness test:

A: Our proposed modified-LIBP protocol:

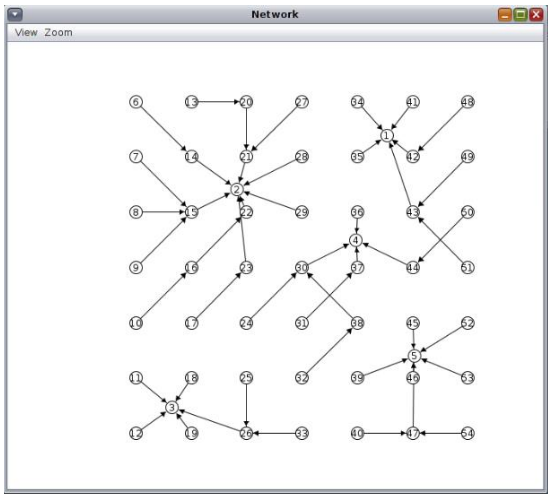


Figure 5-7a: Load balanced network with five sink nodes.

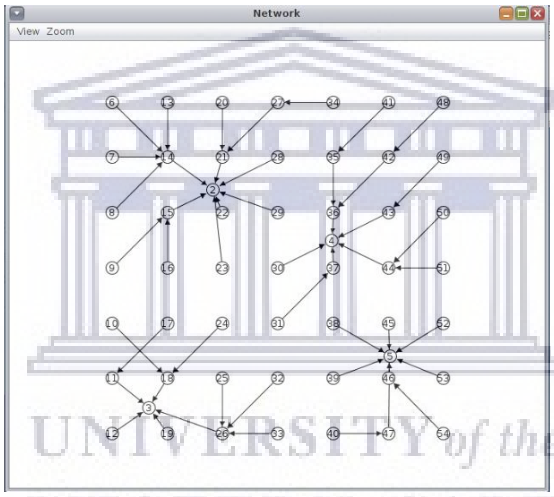


Figure 5-7b: Sink node 1 goes offline due to energy depletion or fault.

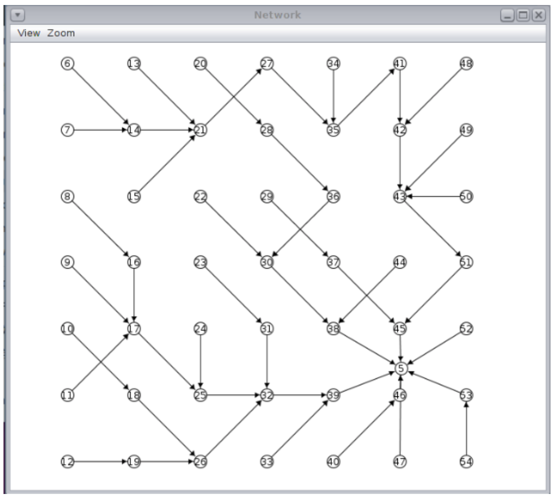


Figure 5-7c: Sink node 1,2,3,4 go offline due to energy depletion or fault.

Figures 5-7a to 5-7c show that as long as at least one sink node remains, the network is able to be recovering from any outage, as all node make use of node 5 as the network sink node.

B: Original LIBP Protocol

Experiments were also done to see what would happen is the sink node goes offline when the original LIBP protocol is used. Figures 5-7d and 5-7e repectively show the results before and after going offline.

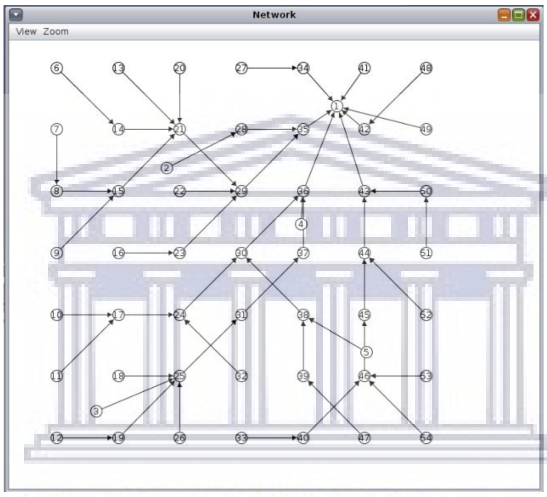


Fig 5-7d: Network shows all nodes making use of node 1 as the sink node.

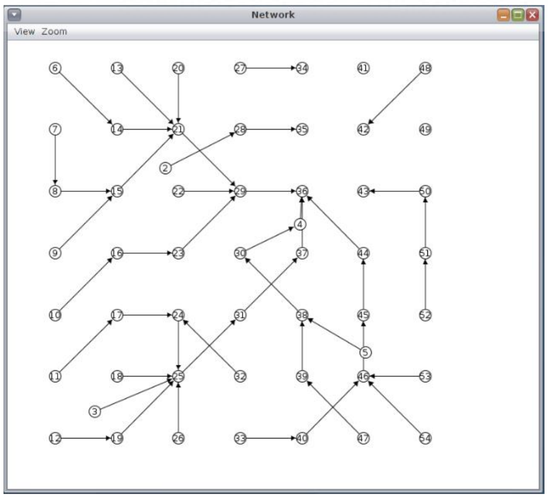
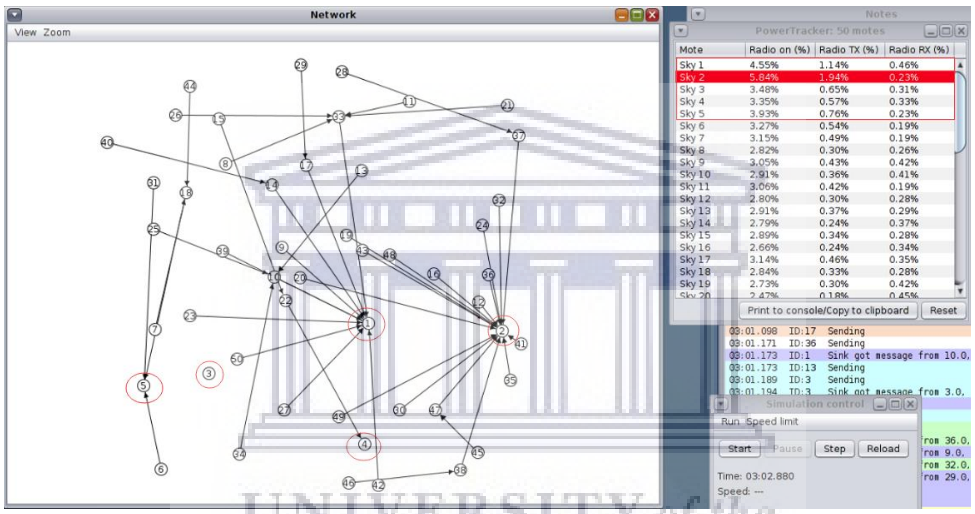


Fig 5-7e: Sink node 1 goes offline and after the network fails to recover.

With the results shown in Figures 5-7a to 5-7e it can be seen that the proposed m-LIBP protocol is more robust to failure than the original LIBP IoT protocol.

Energy consumption test

Experiments were also conducted on a network containing 50 nodes. Radio TX (transmitting) and Radio RX (receive) represent the energy consumption of nodes. A comparison on energy consumption levels was done for single and multi-sink networks (2 to 5 sink nodes). Obtained results are shown in Figures 5.8a-e.

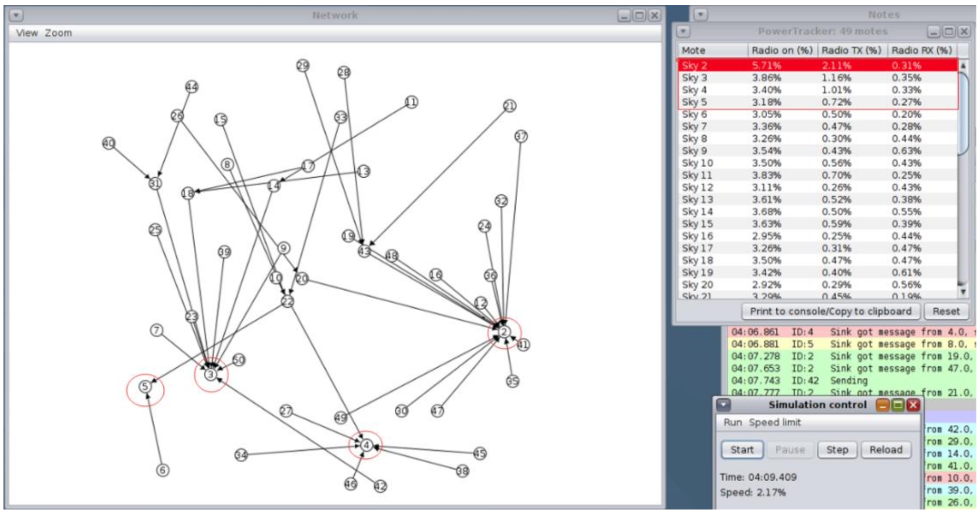


Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	4.55%	1.14%	0.46%
Sky 2	5.84%	1.94%	0.23%
Sky 3	3.48%	0.65%	0.31%
Sky 4	3.35%	0.57%	0.33%
Sky 5	3.93%	0.76%	0.23%
Sky 6	3.27%	0.54%	0.19%
Sky 7	3.15%	0.49%	0.19%
Sky 8	2.82%	0.30%	0.26%
Sky 9	3.05%	0.43%	0.42%
Sky 10	2.91%	0.36%	0.41%
Sky 11	3.06%	0.42%	0.19%
Sky 12	2.80%	0.30%	0.28%
Sky 13	2.91%	0.37%	0.29%
Sky 14	2.79%	0.24%	0.37%
Sky 15	2.89%	0.34%	0.28%
Sky 16	2.66%	0.24%	0.34%
Sky 17	3.14%	0.46%	0.35%
Sky 18	2.84%	0.33%	0.28%
Sky 19	2.73%	0.30%	0.42%
Sky 20	2.47%	0.18%	0.45%

```
03:01.098   ID:17   Sending
03:01.171   ID:36   Sending
03:01.173   ID:1    Sink got message from 10.0,
03:01.173   ID:13   Sending
03:01.189   ID:3    Sending
03:01.194   ID:3    Sink got message from 3.0,
                    rom 36.0,
                    rom 9.0,
                    rom 32.0,
                    rom 29.0,
```

Time: 03:02.880
Speed: ---

Figure 5-8a Energy consumption of five sink nodes.

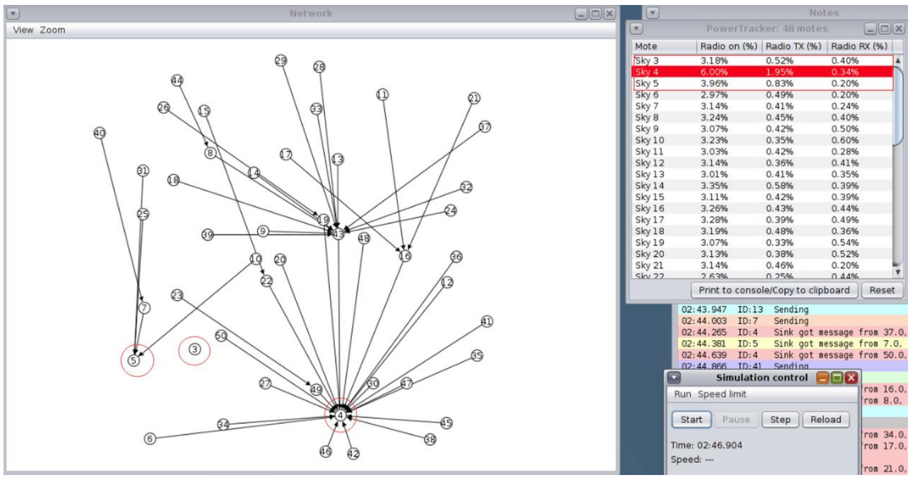


Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 2	5.71%	2.11%	0.31%
Sky 3	3.86%	1.16%	0.35%
Sky 4	3.40%	1.01%	0.33%
Sky 5	3.18%	0.72%	0.27%
Sky 6	3.05%	0.50%	0.20%
Sky 7	3.36%	0.47%	0.28%
Sky 8	3.26%	0.30%	0.44%
Sky 9	3.54%	0.43%	0.63%
Sky 10	3.50%	0.56%	0.43%
Sky 11	3.83%	0.70%	0.25%
Sky 12	3.11%	0.26%	0.43%
Sky 13	3.61%	0.52%	0.38%
Sky 14	3.68%	0.50%	0.55%
Sky 15	3.63%	0.59%	0.39%
Sky 16	2.95%	0.25%	0.44%
Sky 17	3.26%	0.31%	0.47%
Sky 18	3.50%	0.47%	0.47%
Sky 19	3.42%	0.40%	0.61%
Sky 20	2.92%	0.29%	0.56%
Sky 21	3.29%	0.45%	0.19%

```
04:06.861   ID:4    Sink got message from 4.0,
04:06.881   ID:5    Sink got message from 8.0,
04:07.278   ID:2    Sink got message from 19.0,
04:07.653   ID:2    Sink got message from 47.0,
04:07.743   ID:42   Sending
04:07.777   ID:2    Sink got message from 21.0,
                    rom 42.0,
                    rom 29.0,
                    rom 14.0,
                    rom 41.0,
                    rom 10.0,
                    rom 39.0,
                    rom 26.0,
```

Time: 04:09.409
Speed: 2.17%

Figure 5-8b Energy consumption of four sink nodes.



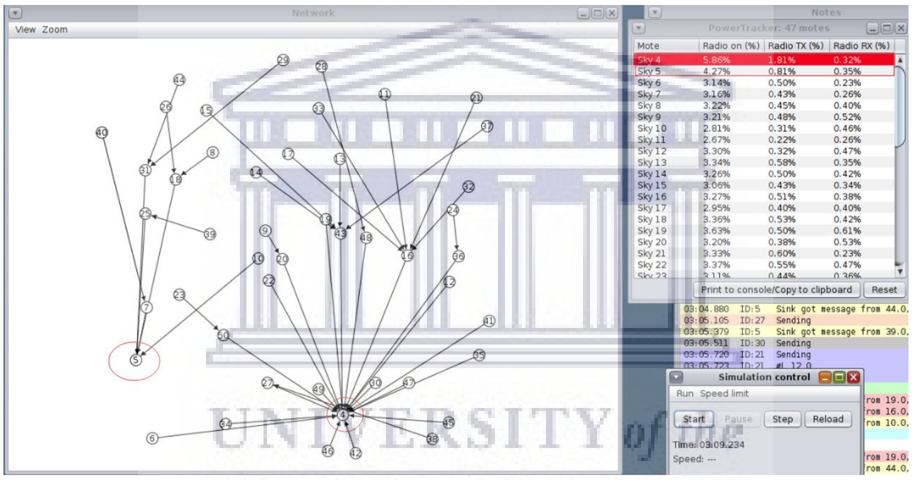
PowerTracker: 48 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 3	3.18%	0.52%	0.40%
Sky 4	6.00%	1.95%	0.34%
Sky 5	3.96%	0.83%	0.20%
Sky 6	2.97%	0.49%	0.20%
Sky 7	3.14%	0.41%	0.24%
Sky 8	3.24%	0.45%	0.40%
Sky 9	3.07%	0.42%	0.50%
Sky 10	3.23%	0.35%	0.60%
Sky 11	3.03%	0.42%	0.28%
Sky 12	3.14%	0.36%	0.41%
Sky 13	3.01%	0.41%	0.35%
Sky 14	3.35%	0.58%	0.39%
Sky 15	3.11%	0.42%	0.39%
Sky 16	3.26%	0.43%	0.44%
Sky 17	3.28%	0.39%	0.49%
Sky 18	3.19%	0.48%	0.36%
Sky 19	3.07%	0.33%	0.54%
Sky 20	3.13%	0.38%	0.52%
Sky 21	3.14%	0.46%	0.20%
Sky 22	2.63%	0.25%	0.44%

```
02:43.947   ID:13   Sending
02:44.003   ID:7    Sending
02:44.265   ID:4    Sink got message from 37.0,
02:44.381   ID:5    Sink got message from 7.0,
02:44.639   ID:4    Sink got message from 50.0,
02:44.866   ID:41   Sending
```

```
Time: 02:46.904
Speed: ---
```

Figure 5-8c Energy consumption of three sink nodes.



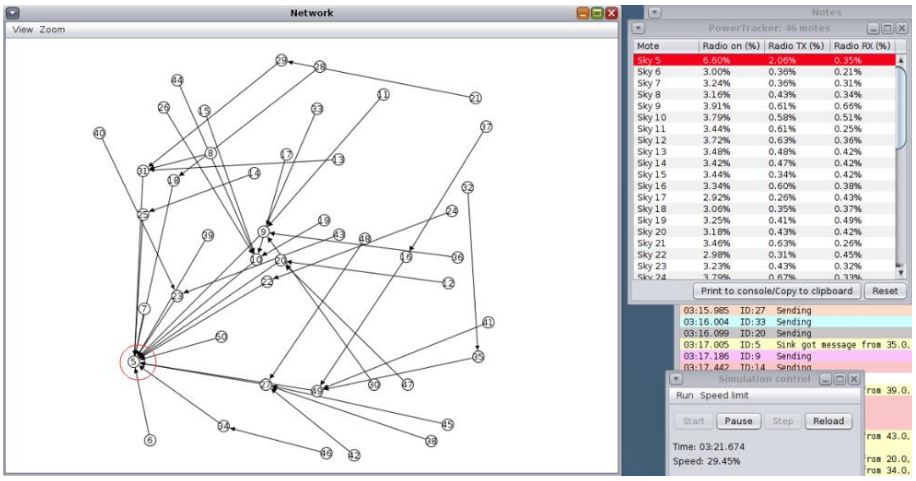
PowerTracker: 47 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 4	5.86%	1.81%	0.32%
Sky 5	4.27%	0.81%	0.35%
Sky 6	3.14%	0.50%	0.23%
Sky 7	3.16%	0.43%	0.26%
Sky 8	3.22%	0.45%	0.40%
Sky 9	3.21%	0.48%	0.52%
Sky 10	2.81%	0.31%	0.46%
Sky 11	2.67%	0.22%	0.26%
Sky 12	3.30%	0.32%	0.47%
Sky 13	3.34%	0.58%	0.35%
Sky 14	3.26%	0.50%	0.42%
Sky 15	3.06%	0.43%	0.34%
Sky 16	3.27%	0.51%	0.38%
Sky 17	2.95%	0.40%	0.40%
Sky 18	3.36%	0.53%	0.42%
Sky 19	3.63%	0.50%	0.61%
Sky 20	3.20%	0.38%	0.53%
Sky 21	3.33%	0.60%	0.23%
Sky 22	3.37%	0.55%	0.47%
Sky 23	3.11%	0.44%	0.36%

```
03:04.880   ID:5    Sink got message from 44.0,
03:05.105   ID:27   Sending
03:05.379   ID:5    Sink got message from 39.0,
03:05.511   ID:30   Sending
03:05.720   ID:21   Sending
03:05.723   ID:21   #L 12.0
```

```
Time: 03:09.234
Speed: ---
```

Figure 5-8d Energy consumption of two sink nodes.



PowerTracker: 46 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 5	6.60%	2.06%	0.35%
Sky 6	3.00%	0.36%	0.21%
Sky 7	3.24%	0.36%	0.31%
Sky 8	3.16%	0.43%	0.34%
Sky 9	3.91%	0.61%	0.66%
Sky 10	3.79%	0.58%	0.51%
Sky 11	3.44%	0.61%	0.25%
Sky 12	3.72%	0.63%	0.36%
Sky 13	3.48%	0.48%	0.42%
Sky 14	3.42%	0.47%	0.42%
Sky 15	3.44%	0.34%	0.42%
Sky 16	3.34%	0.60%	0.38%
Sky 17	2.92%	0.26%	0.43%
Sky 18	3.06%	0.35%	0.37%
Sky 19	3.25%	0.41%	0.49%
Sky 20	3.18%	0.43%	0.42%
Sky 21	3.46%	0.63%	0.26%
Sky 22	2.98%	0.31%	0.45%
Sky 23	3.23%	0.43%	0.32%
Sky 24	3.79%	0.67%	0.33%

```
03:15.985   ID:27   Sending
03:16.004   ID:33   Sending
03:16.099   ID:20   Sending
03:17.005   ID:5    Sink got message from 35.0,
03:17.186   ID:9    Sending
03:17.442   ID:14   Sending
```

```
Time: 03:21.674
Speed: 29.45%
```

Figure 5-8e Energy consumption of one sink nodes.

From the perspective of the IoT terminal layer, nodes' energy consumption is the goal that routing protocol needs to achieve. It can be seen that the highest energy consumed by the sink nodes from Figure 5-8a-d are respectively 5.84%, 5.7%, 6.00%, 5.86%. These are lower than the energy consumption of 6.60% recorded with the original LIBP (1 sink node) in Figure 5-8e was used. Similarly, on Table 5-3, the average energy consumed when using multiple sink node in each figure are respectively 4.23%, 3.92%, 4.38%, 5.07%. These are again lower than the average of the original LIBP at 6.60%. Comparing the results, the proposed multi-sink LIBP used by IoT-FCM shows lower energy consumption versus the original LIBP. This in turn implies better battery life of sink nodes.

Table 5-3: Comparison of Energy consumption using multiple sink nodes

	Energy consumption of 5 sink nodes	Energy consumption of 4 sink nodes	Energy consumption of 3 sink nodes	Energy consumption of 2 sink nodes	Energy consumption of 1 sink nodes
Sink node 1:	4.55%	**5.71%**	3.18%	**5.86%**	**6.60%**
Sink node 2:	**5.84%**	3.86%	**6.00%**	4.27%	none
Sink node 3:	3.48%	3.40%	3.96%	none	none
Sink node 4:	3.35%	3.28%	none	none	none
Sink node 5:	3.93%	none	none	none	none
Average:	4.23%	3.92%	4.38%	5.07%	6.6%

In order to further test the energy consumption, we set up another scenario case, where we put the multi-sink LIBP with 3 sink nodes in three clusters, with each cluster having 1 sink node. The original number of nodes in each cluster were randomly set to 10, 21 and 17 respectively. Then we adjusted the number of nodes in each cluster to see the energy consumption situations of the highest energy consumption node and the average energy consumption The simulation results of this process shows that moving nodes from different cluster is able to decrease the highest energy consumption, which are summarized on Table 5-4.

Table 5-4: Energy balance moving nodes from different sink node cluster

	Sink 1 Sink 2 Sink 3		Sink 1 Sink 2 Sink 3		Sink 1 Sink 2 Sink 3	
	10 21 17		15 16 17		16 16 16	
	Highest	Average	Highest	Average	Highest	Average
Sink 1	4.18%	2.09%	4.77%	2.51%	4.77%	2.58%
Sink 2	**5.54%**	**2.89%**	4.83%	2.66%	**4.83%**	**2.66%**
Sink 3	4.96%	2.44%	**4.85%**	**2.71%**	4.79%	2.61%

We also tested the recovery time by making the sink node 1 offline, and compared the recovery time of different number of sink nodes to demonstrate the robustness of the multi-sink LIBP. The results of these are shown on Table 5-5.

Table 5-5: Comparison of longest distance and recovery time using multiple sink nodes

	5 sink nodes					4 sink nodes				3 sink nodes			2 sink nodes		1 sink nodes
Sink node number	1	2	3	4	5	1	2	3	4	1	2	3	1	2	1
Distance	2	2	0	2	2	2	1	2	2	2	3	0	2	2	3
Recovery time	**15.183s**	none				**18.768s**	none			**27.328s**	none		**34.812s**	none	**infinity**

5.5 Conclusion:

Fog computing is a new architectural model, derivative from Cloud Computing. It adopts the design of decentralized network structure, streamlines the process of data aggregation transmission, which can greatly reduce network delay, reduce bandwidth requirements and improve network security. Processing all data from the Internet of Things (IoT) and sensors by Cloud Computing is relatively inefficient, but Fog Computing can solve this problem as Fog Computing technology brings the Cloud closer to the clients.

Combining the Internet of Things and Fog Computing, we proposed the IoT-based Fog computing model and describe this model in layers. In the Fog

Computing layer part of our model, we use fog nodes to handle specific tasks from IoT layer, the IoT-FCM moved users 38% closer to the Fog node for Fog-oriented Max-Min and 55% for the traditional Max-Min. While with respect to energy, IoT-FCM conserved an average of 150KWh more energy versus the other algorithms. For the other part of the model, which is the terminal layer; IoT-FCM modified the LIBP protocol by adding multiple sinks. Performance evaluations were done using Cooja on Contiki and obtained results, which show that the modified LIBP with its use of multiple sink nodes was more robust and tolerant to node failure and was also more energy conservatory. Of significant note in this work is that the two layers were simulated on different environments – CloudSim and Cooja.

Chapter 6: Conclusion and Future Work

6.1 Retrospective

With the development of Cloud Computing and the maturity of related technologies, the user base of Cloud Computing as a platform for commercial and personal use has grown tremendously. The number of tasks submitted by users and the scale of these tasks are becoming larger. Users expect the Cloud to always be on, always available and accessible from anywhere and at any time. Despite the general perception of the Cloud being an unlimited pool of resources, in truth, Cloud resources are finite and Cloud Service Providers (CSP) often struggle with accommodating the ever increasing heterogeneous workloads submitted by Cloud users. This issue has fuelled studies on techniques of effectively allocating workloads to resources.

This thesis has discussed four major problems related to Cloud Computing. Firstly, it looked at the problem of fair distribution of profit in Cloud federation and concluded that most related works only consider maximizing interests as incentive targets, but does not fully consider the influence and role of profit distribution among the federation members. Secondly, most existing research on task scheduling in the Cloud Computing environment are not able to guarantee the overall QoS targets of both users and CPS, while maximizing overall benefits. Thirdly, with respect to research on consolidation of virtual machines, most ignored the important factor of virtual machine migration overhead when selecting VMs to be migrated across host servers. Lastly, for the communication research between Internet of Things and Cloud Computing, the influence of bandwidth, delay and energy consumption caused by geographical location should be given more attention.

In view of the above mentioned, this thesis conducted an in-depth research into resource allocation in Cloud Computing and made the contributions summarized as follows:

Resource management and profit distribution: This thesis analysed the necessity of Cloud federation, and its key technologies. Fair profit distribution was identified as an incentive for CSPs to join a Cloud federation, hence a Cloud federation model based on Shapley Value for fair profit distribution strategy (SVPDS) was proposed. The model is a dynamic benefit allocation mechanism, which enables the profit distribution between Cloud providers to be dynamically adjusted according to the changes of federation members and the tasks it received. Results of simulations showed that compared with the traditional method of benefit distribution of To Each According to Their Contribution (TEATC), SVPDS can comprehensively assess the contribution degree of each CSP in the federation and always fairly distribute profit among them.

Resource management and task scheduling problem: The purpose of Cloud Computing is to provide users with a comprehensive service. However, while providing such services, the quality of service is often compromised; especially from the perspective of users as the definition of quality varies across users. With this in mind, this thesis proposed a model that considers multi-QoS target constraints task scheduling. The model modifies the classic Differential Evolution algorithm and uses it to convert the multi QoS requirements problem to a single QoS problem. The modification also reduces the probability of the DE being trapped in a local optimal solution. Compared with the traditional method, the proposed algorithm is more stable and has a better comprehensive performance in adhering to quality of service and minimizing task makespan.

Resource management, VM migration and Consolidation: This thesis has contributed in the area of virtual machine consolidation by introducing a model that is migration overhead aware. The proposed model controls unnecessary virtual machine migration by its overhead awareness, and combined it with both cooperative federated Cloud Computing and competitive federated Cloud Computing. This developed model is able to reduce migration overhead during virtual machine consolidation by minimizing unnecessary migrations. Results of experimental simulations show that the model can indeed reduce the virtual machine migration overhead as well as reduce the energy exerted during VM migration and consolidation in both cooperative and competitive Cloud federations.

Resource allocation in IoT and Fog Networks: This thesis proposed an IoT-based Fog Computing model which consists of two layers: the IoT layer and the Fog layer. At the IoT layer, data is gathered and uploading to the Fog. This work modified the Least Interference Beaconing Protocol making it more robust and energy efficient. This was achieved by introducing multiple sink nodes. The Fog layer receives and processes the data from the IoT layer. This thesis modified the classic GA to optimize the task allocation to Fog nodes. Results of experiments conducted showed that the models proposed by this thesis yielded better performance in terms of robustness and energy control at the IoT layer; yet equally efficient with regards delay, distance and energy consumption between the two layers.

6.2 Perspective

This work can be strengthened in many different directions offering many opportunities for future works. Some of these considerations include:

- Blockchain technology in Cloud federation. In addition to fair profit distribution and benefit maximizing, future work could look into other factors that can encourage CSP participation in Cloud federation. A billing system that can securely and immutably store every resource usage transaction could be considered. Blockchain technology can pave the way for such a financial billing system using encrypted blocks to store data in a common ledger. The use of Blockchain technology to support the development of Cloud federation should be a very promising research topic.
- Technical standards normalization. Despite the numerous research works

on going, there is still a lack of globally accepted standard in Cloud Computing. Issues of vendor lock in and incompatibility are still prevalent, due to the use of proprietory technologies by various CSPs. The lack of technical standards has without doubt hindered the development of Cloud Computing technology. Developing a uniform and interoperable platform for the Cloud could be another interesting area for future research works.

- The gap between theoretical and practical. Virtual machine consolidation achieves energy savings by reducing the number of compute nodes used. In the virtual machine consolidation research conducted in this thesis, there is an implicit assumption that multiple virtual machines running on the same computing node do not affect each other. However, in reality, when multiple virtual machines run on the same computing node, they are not completely isolated from each other. A number of researchers have reported that multi-tenant VMs can compete for shared resources, which may cause performance interference. Furthermore, VM migration and consolidation in cooperative Cloud federation might be more complex to achieve in practice than in theory. Proffering solutions to these VM consolidation related problems could also be directions for extending this thesis.

REFERENCES

1. Cusumano M. Cloud computing and SaaS as new computing platforms[J]. Communications of the ACM, 2010, 53(4): 27-29.

2. Manias E., B.F., *A component-based middleware for hybrid grid/cloud computing platforms[J].* . Concurrency and computation: practice & experience, 2012. 24(13): 1461-1477

3. Armbrust M., F.A., Griffith R., et al. , *A view of cloud computing[J].* . Communications of the ACM, 2010, 53(4): 50-58.

4. Luis M. V., L.R.M., Juan C., et al. , *A break in the clouds: towards a cloud definition[J].* . ACM SIGCOMM computer communication review, 2009. 39(1): 50-55

5. Beloglazov A, Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers[J]. Concurrency and Computation: Practice and Experience, 2012, 24(13): 1397-1420.

6. Foster I., Z.Y., Raicu L., et al. , *Cloud Computing and grid computing 360-degree compared[A]. Proceedings of the 2008 grid computing environments workshop[C]*. 2008.

7. Yan., L., *Network I/O virtualization for cloud computing[J].* IT professional, 2010. 12(5): 36-41

8. Ghosh A., A.I., *In cloud computing we trust – but should we?[J].* IEEE security & privacy, 2010. 8(6): 14-16

9. Brumec S., V.N., *Cost effectiveness of commercial computing clouds[J].* Information systems, 2013. 38(4): 495-508

10. Liu Ke, J.H., Chen Jinjun, et al. , *A compromised-time-cost scheduling algorithm in swin De W-C for instance-intensive cost-constrained workflows on a cloud computing platform[J].* International journal of high performance computing applications, 2010. 24(4): 445-456

11. Fard, H.M., R. Prodan, and T. Fahringer, *A truthful dynamic workflow scheduling mechanism for commercial multicloud environments.* IEEE Transactions on Parallel and Distributed systems, 2013. 24(6): p. 1203-1212.

12. Su, S., et al., *Cost-efficient task scheduling for executing large programs in the cloud.* Parallel Computing, 2013. 39(4-5): p. 177-188.

13. Moreno-Vozmediano, R., R.S. Montero, and I.M. Llorente, *Key challenges in cloud computing: Enabling the future internet of services.* IEEE Internet Computing, 2013. 17(4): p. 18-25.

14. Delimitrou, C. and C. Kozyrakis, *Qos-aware scheduling in heterogeneous datacenters with paragon.* ACM Transactions on Computer Systems (TOCS), 2013. 31(4): p. 12.

15. Wang, W.-J., et al., *Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments.* The Journal of Supercomputing, 2013. 66(2): p. 783-811.

16. Grozev, N. and R. Buyya, *Inter‐Cloud architectures and application brokering: taxonomy and survey.* Software: Practice and Experience, 2014. 44(3): p. 369-390.

17. Giacobbe, M., et al., *Towards energy management in Cloud federation: A survey in the perspective of future sustainable and cost-saving strategies.* Computer Networks, 2015. 91: p. 438-452.

18. Goiri, I., Guitart, J. and Torres, J., *Characterizing cloud federation for enhancing providers' profit. In Cloud Computing (CLOUD)*, in *2010 IEEE 3rd International Conference on (pp. 123-130). IEEE*. 2010, July.

19. Fazio, M., et al. *How to enhance cloud architectures to enable cross-federation: Towards interoperable storage providers*. in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. 2015. IEEE.

20. El Zant, B., Amigo, I. and Gagnaire, M., *Federation and revenue sharing in cloud computing environment. In Cloud Engineering (IC2E)*, in *In Cloud Engineering (IC2E), 2014 IEEE International Conference on (pp. 446-451). IEEE*. 2014.

21. Okuhara, M., T. Shiozaki, and T. Suzuki, *Security architecture for cloud computing.* Fujitsu Sci. Tech. J, 2010. 46(4): p. 397-402.

22. Ranganathan, P., et al., *Ensemble-level Power Management for Dense Blade Servers.* SIGARCH Comput. Archit. News, 2006. 34(2): p. 66-77.

23. Horvath, T., et al., *Dynamic Voltage Scaling in Multitier Web Servers with End-to-End Delay Control.* IEEE Transactions on Computers, 2007. 56(4): p. 444-458.

24. Alboaneen, D.A., B. Pranggono, and H. Tianfield, *Energy-Aware Virtual Machine Consolidation for Cloud Data Centers*, in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. 2014, IEEE Computer Society. p. 1010-1015.

25. Wolke, A. and C. Pfeiffer. *Improving Enterprise VM Consolidation with High-Dimensional Load Profiles*. in *2014 IEEE International Conference on Cloud Engineering*. 2014.

26. Perumal, V. and S. Subbiah, *Power-conservative server consolidation based resource management in cloud.* International Journal of Network Management, 2014. 24(6): p. 415-432.

27. Ahmad, R.W., et al., *A survey on virtual machine migration and server consolidation frameworks for cloud data centers.* Journal of Network and Computer Applications, 2015. 52: p. 11-25.

28. Laszewski, G.v., et al. *Power-aware scheduling of virtual machines in DVFS-enabled clusters*. in *2009 IEEE International Conference on Cluster Computing and Workshops*. 2009.

29. Ding, Y., et al., *Energy efficient scheduling of virtual machines in cloud with deadline constraint.* Future Generation Computer Systems, 2015. 50: p. 62-74.

30. Kamga, C.M., G.S. Tran, and L. Broto, *Extended scheduler for efficient frequency scaling in virtualized systems.* SIGOPS Oper. Syst. Rev., 2012. 46(2): p. 28-35.

31. Dargie, W. *Estimation of the cost of VM migration*. in *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*. 2014.

32. W. Voorsluys, J.B., S. Venugopal, et al. , *Cost of virtual machine live migration in clouds: A performance evaluation.*, in *Proceedings of the Internationall Conference on Cloud Computing*. 2009: Beijing.

33. F. Xu, L.F., Jin H, et al. , *Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions.* Proceedings of the IEEE, 2014. 102(1): 11-31

34. *Bonomi F, Milito R, Natarajan P, et al. Fog computing: A platform for internet of things and analytics[M]//Big data and internet of things: A roadmap for smart environments. Springer, Cham, 2014: 169-186.*

35. Zhao, M., et al., *A comprehensive study of RPL and P2P-RPL routing protocols: Implementation, challenges and opportunities.* Peer-to-Peer Networking and Applications, 2017. 10(5): p. 1232-1256.

36. Pawel Michalak, T., et al., *Efficient Computation of the Shapley Value for Game-Theoretic Network Centrality.* arXiv preprint arXiv:1402.0567, 2014.

37. Buyya R, Y.S., Venugopal S, et al. , *Cloud Computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility [J].* Future Generation Computer Systems, 2009. 25 (6): 599-616.

38. Mashayekhy, L., M.M. Nejad, and D. Grosu, *Cloud federations in the sky: Formation game and mechanism.* IEEE Transactions on Cloud Computing, 2015. 3(1): p. 14-27.

39. Bates, G.H.T. and P. Smith. *Cidr report-http://www.cidrreport.org/as2.0/ Technical report*. 2015. CIDR.

40. Kertesz, A., *Characterizing cloud federation approaches*, in *Cloud Computing*. 2014, Springer. p. 277-296.

41. Shapley, L., *"A value for n-person games," in Contributions to the Theory of Games II, H. W. Kuhn and A. W. Tucker, Eds. Princeton Univ. Press*. 1953,.

42. Bessis, N., et al. *An architectural strategy for meta-scheduling in Inter-clouds*. in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. 2012. IEEE.

43. Assis, M.R. and L.F. Bittencourt, *A survey on cloud federation architectures: Identifying functional and non-functional properties.* Journal of Network and Computer Applications, 2016. 72: p. 51-71.

44. *Li H. Cloud federation as a service: U.S. Patent 8,924,569[P]. 2014-12-30.*

45. Celesti, A., et al. *Three-phase cross-cloud federation model: The cloud sso authentication*. in *Advances in Future Internet (AFIN), 2010 second international conference on*. 2010. IEEE.

46. Ray, B., et al., *Quality and Profit Assured Trusted Cloud Federation Formation: Game Theory Based Approach.* IEEE Transactions on Services Computing, 2018.

47. Saeed Araban, L.S., *Measuring Quality of Srevice for Contract Aware Web-Srevice[J].* Proceedings of the 1st Australian Work- shop on Engineering Service-Oriented System,Melbourne,Australia, 2004. 54-56.

48. Ran, S., *A model for web services discovery with QoS.* ACM Sigecom exchanges, 2003. 4(1): p. 1-10.

49. Mani, A. and A. Nagarajan, *Understanding quality of service for Web services–Improving the performance of your Web services, 2002*. 2010.

50. Dhuria, S., A. Gupta, and R. Singla, *Comparison of Revenue Sharing Mechanisms in Cloud Federation.* 2018.

51. Toosi, A.N., Calheiros, R.N., Thulasiram, R.K. and Buyya, R., *Resource provisioning policies to increase iaas provider's profit in a federated cloud environment.*, in *In High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on (pp. 279-287). IEEE.* 2011, September. .

52. Hassan, M.M., Abdullah-Al-Wadud, M., Almogren, A., Song, B. and Alamri, A. , *Energy-aware resource and revenue management in federated cloud: a game-theoretic approach.* . IEEE Systems Journal, 2017. 11(2), pp.951-961.

53. Tang, L.a.C., H., *Double auction mechanism for request outsourcing in cloud federation*, in *In Communication Workshop (ICCW), 2015 IEEE International Conference on (pp. 1889-1894). IEEE.* 2015, June.

54. A. A. Young, B.N.C., A. C. Snoeren, and A. Vahdat, *"Resource allocation in federated distributed computing infrastructures," in Proc. OASIS Workshop.* 2004.

55. Dramitinos M, Stamoulis G D, Courcoubetis C. An auction mechanism for allocating the bandwidth of networks to their users[J]. Computer Networks, 2007, 51(18): 4979-4996.

56. Walrand, L.H.a.J., *"Pricing and revenue sharing strategies for Internet service providers" in Proc. IEEE INFOCOM*. 2005.

57. Walrand, R.J.a.J., *"An efficient Nash-implementation mechanism for network resource allocation,"   Automatica*. 2010.

58. Androulaki M, Frangedaki E, Antoniadis P. Optimization of public spaces through network potentials of communities[J]. Procedia Manufacturing, 2020, 44: 294-301.

59. Norman, G., *Non-Cooperative Game*, in *Dictionary of Industrial Organization*. 2014, Edward Elgar Publishing Limited.

60. Mashayekhy, L. and D. Grosu. *A coalitional game-based mechanism for forming cloud federations*. in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*. 2012. IEEE Computer Society.

61. Winter, E., *Handbook of Game Theory with Economic Applications, Chapter 53 The shapley value. Hebrew University of Jerusalem, Volume 3, 2002.*

62. Gregory and Stuart, P.a.R.C.E.S.i.t.T.-F.S.-W.C.P.p.I.

63. Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms[J]. Software: Practice and experience, 2011, 41(1): 23-50.

64. Belalem, G., F.Z. Tayeb, and W. Zaoui. *Approaches to improve the resources management in the simulator CloudSim*. in *International Conference on Information Computing and Applications*. 2010. Springer.

65. Subrata, R., A.Y. Zomaya, and B. Landfeldt, *A cooperative game framework for QoS guided job allocation schemes in grids.* IEEE Transactions on Computers, 2008. 57(10): p. 1413-1422.

66. I. Ch. Paschalidis and Y. Liu, *"Pricing in multiservice loss networks: static pricing, asymptotic optimality, and demand substitution effects,".* IEEE/ACM ToN, 2002. vol. 10, no. 3.

67. Laili Yuanjun, T.F., Zhang Lin, et al. , *A study of optimal allocation of computing resources in cloud manufacturing systems[J].* The international journal of advanced manufacturing technology, 2012. 63(5-8): 671-690.

68. Jang E. Y., K.H.J., *Rule-based cloud RBAC model for flexible resource allocation in cloud computing service[J].* Information-an international interdisciplinary journal, 2010. 13(5): 1653-1666

69. Han Yanbo, S.J., Wang Guiling, et al. , *A cloud-based BPM architecture with user-end distribution of non-compute-intensive activities and sensitive data[J].* Journal of computer science and technology, 2010. 25(6): 1157-1167

70. Lin Weiwei, L.C., Wang J. Z., et al. , *Bandwidth-aware divisible task scheduling for cloud computing[J].* . Software: practice and experience, 2014. 44(2): 163-174.

71. Hirai T., M.H., Kasahara S., et al. , *Performance analysis of large-scale parallel-distributed processing with backup tasks for cloud computing[J]*. Journal of industrial and management optimization, 2014. 10(1): 113-129

72. Pedersen J. M., R.M.T., Dubalski B., et al. , *Using latency as a QoS indicator for a global cloud computing services[J]*. Concurrency and computation practice and experience, 2013. 25(18): 2488-2500

73. Pop F., C.V., Bessis N., et al. , *Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments[A]*. Proceedings of the 27thinternational conference on advanced information networking and applications workshops[C], 2013. 772-776.

74. Stieninger, M., et al., *Factors influencing the organizational adoption of cloud computing: a survey among cloud workers.* Determinants of analytics-based managerial decision-making, 2018.

75. Hameed, A., et al., *A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems.* Computing, 2016. 98(7): p. 751-774.

76. Kaikai, S., X. Wensheng, and L. Jianyong, *Research on Mass Manufacturing Resource Sensory Data Management Based On Hadoop.* Key Engineering Materials, 2016. 693.

77. Pillai, P.S. and S. Rao, *Resource allocation in cloud computing using the uncertainty principle of game theory.* IEEE Systems Journal, 2016. 10(2): p. 637-648.

78. Anastasi, G.F., et al., *QBROKAGE: A Genetic Approach for QoS Cloud Brokering*, in *Proceedings of the 2014 IEEE International Conference on Cloud Computing*. 2014, IEEE Computer Society. p. 304-311.

79. Bruneo, D., *A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems.* IEEE Transactions on Parallel and Distributed Systems, 2014. 25(3): p. 560-569.

80. Singh, S. and I. Chana, *QRSF: QoS-aware resource scheduling framework in cloud computing.* The Journal of Supercomputing, 2015. 71(1): p. 241-292.

81. LI Tao-shen, Z.X.-x., *Evolutionary Game Scheduling Algorithm for Differentiated Services under Cloud Computing [J].* Journal of Beijing University of Posts and Telecommunications, 2013. 36 (1): 41-45.

82. Samanta, A. and S. Misra, *Dynamic Connectivity Establishment and Cooperative Scheduling for QoS-Aware Wireless Body Area Networks.* IEEE Transactions on Mobile Computing, 2018.

83. Kumar, N., et al., *Bayesian cooperative coalition game as-a-service for RFID-based secure QoS management in mobile cloud.* IEEE Transactions on Emerging Topics in Computing, 2016.

84. Daniel Paranhos da Silva, W.C., Francisco Vilar Brasileiro. , *Trading Cycles for Information: Using Replication to Schedule Bag-of-Tasks Applications on Computational* Euro-Par 2003 Parallel Processing, 2003 pp 169-180.

85. Oprescu A., K.T., *Bag-of-tasks scheduling under budget constraints[A].* Proceedings of the 2ndIEEE international conference on cloud computing technology and science[C], 2010. 351-359

86. Wei Yu, L.Y., LinHung Yu, Wei, *Dynamic Auction Mechanism for Cloud Resource Allocation[A].* CCGRID '10 Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing[C], 2010. May 17 - 20, 2010.

87. Wolski R., S.N.T., Hayes J., *The network weather service: a distributed resource performance forecasting service for meta-computing[J]*. Future generation computer systems, 1999. 15(5-6): 757-768.

88. Randies, M., D. Lamb, and A. Taleb-Bendiab. *A comparative study into distributed load balancing algorithms for cloud computing*. in *Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on*. 2010. IEEE.

89. Aslam, S. and M.A. Shah. *Load balancing algorithms in cloud computing: A survey of modern techniques*. in *Software Engineering Conference (NSEC), 2015 National*. 2015. IEEE.

90. Fujimoto, N. and K. Hagihara. *A comparison among grid scheduling algorithms for independent coarse-grained tasks*. in *Applications and the Internet Workshops, 2004. SAINT 2004 Workshops. 2004 International Symposium on*. 2004. IEEE.

91. Freund, R.F., et al. *Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet*. in *Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh*. 1998. IEEE.

92. Wu, M.-Y., W. Shu, and H. Zhang. *Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems*. in *hcw*. 2000. IEEE.

93. Maheswaran, M., et al. *Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems*. in *Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth*. 1999. IEEE.

94. Lai, G.-J. *A novel task scheduling algorithm for distributed heterogeneous computing systems*. in *International Workshop on Applied Parallel Computing*. 2004. Springer.

95. Qiu, M., et al., *Phase-change memory optimization for green cloud with genetic algorithm.* IEEE Transactions on Computers, 2015. 64(12): p. 3528-3540.

96. Keshanchi, B., A. Souri, and N.J. Navimipour, *An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing.* Journal of Systems and Software, 2017. 124: p. 1-21.

97. Verma, A. and S. Kaushal, *Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud.* International Journal of Grid and Utility Computing, 2014. 5(2): p. 96-106.

98. Gai, K., M. Qiu, and H. Zhao, *Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing.* IEEE transactions on cloud computing, 2016.

99. Zuo, L., et al., *A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing.* IEEE Access, 2015. 3: p. 2687-2699.

100. Dam, S., et al., *An ant colony based load balancing strategy in cloud computing*, in *Advanced Computing, Networking and Informatics-Volume 2*. 2014, Springer. p. 403-413.

101. Hu, Y., et al., *Cloud manufacturing resources fuzzy classification based on genetic simulated annealing algorithm.* Materials and Manufacturing Processes, 2017. 32(10): p. 1109-1115.

102. Alkayal, E.S., N.R. Jennings, and M.F. Abulkhair. *Survey of task scheduling in cloud computing based on particle swarm optimization*. in *Electrical and Computing Technologies and Applications (ICECTA), 2017 International Conference on*. 2017. IEEE.

103. Awad, A., N. El-Hefnawy, and H. Abdel_kader, *Enhanced particle swarm optimization for task scheduling in cloud computing environments.* Procedia Computer Science, 2015. 65: p. 920-929.

104. Zhang, Y., S. Wang, and G. Ji, *A comprehensive survey on particle swarm optimization algorithm and its applications.* Mathematical Problems in Engineering, 2015. 2015.

105. Verma, A. and S. Kaushal. *Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud*. in *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*. 2014. IEEE.

106. De Falco, I., et al. *Improving search by incorporating evolution principles in parallel tabu search*. in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. 1994. IEEE.

107. Chow, K.-W. and B. Liu. *On mapping signal processing algorithms to a heterogeneous multiprocessor system*. in *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*. 1991. IEEE.

108. Robert, C. and G. Casella, *Monte Carlo statistical methods*. 2013: Springer Science & Business Media.

109. Sun, J., Q. Zhang, and E.P. Tsang, *DE/EDA: A new evolutionary algorithm for global optimization.* Information Sciences, 2005. 169(3-4): p. 249-262.

110. Zhao, X., X.-S. Gao, and Z.-C. Hu, *Evolutionary programming based on non-uniform mutation.* Applied Mathematics and Computation, 2007. 192(1): p. 1-11.

111. Lipowski, A. and D. Lipowska, *Roulette-wheel selection via stochastic acceptance.* Physica A: Statistical Mechanics and its Applications, 2012. 391(6): p. 2193-2196.

112. J. O. Gutierrez-Garcia, A.R.-N., *Collaborative agents for distributed load management in cloud data centers using live migration of virtual machines[J].* IEEE Transactions on Service Computing,, 2015. 8(6): 916-929

113. N. J. Kansal, I.C., *Energy-aware virtual machine migration for cloud computing-a cirefly optimization approach[J].* Journal of Grid Computing, 2016. 14(2): 327-345

114. V. Soundararajan, J.M.A., *The impact of management operations on the virtualized datacenter[C].* in *Proceedings of the ACM International Symposium on Computer Architecture*. 2010: Saint-Malo.

115. H. Liu, H.J., C. Z. Xu, X. Liao. , *Performance and energy modeling for live migration of virtual machines.* Cluster Computing, 2013. 16: 249 - 264

116. H. Liu, H.J., X. Liao, et al. , *Live virtual machine migration via asynchronous replication and state synchronization.* IEEE Transactions on Parallel and Distributed Systems, 2011. 22(12): 1986-1999

117. A. Beloglazov, J.A., R. Buyya. , *Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing.* Future generation computer systems, 2012. 28(5): 755-768

118. F. Xu, L.F., Jin H, et al, *Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions.* Proceedings of the IEEE, 2014. 102(1): 11-31

119. N. J. Kansal, I.C., *Energy-aware virtual machine migration for cloud computing-a cirefly optimization approach.* Journal of Grid Computing, 2016. 14(2): 327-345

120. T. C. Ferreto, M.A.S.N., R. N. Calheiros, et al. Server consolidation with migration control for virtualized data centers[J]. Future Generation Computer Systems, 2011, 27(8): 1027-1034, *Server consolidation with migration control for virtualized data centers[J].* Future Generation Computer Systems, 2011. 27(8): 1027-1034

121. M. Marzolla, O.B., F. Panzieri, *Server consolidation in clouds through gossiping[C].* in *Proceedings of the International Symposium on World of Wireless, Mobile and Multimedia Networks*. 2011 Lucca.

122. A. Verma, P.A., A. Neogi, *pMapper: power and migration cost aware application placement in virtualized systems[C]*, in *Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference*. 2008: Leuven.

123. E. Feller, C.M., *Autonomous and energy-aware management of large-scale cloud infrastructures*, in *Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshops and Ph D Forum*. 2012: Shanghai.

124. S. Chen, J.W., Z. H. Lu, *A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness[C].* in *Proceedings of the 12th International Conference on Computer and Information Technology*. 2012: Chengdu.

125. Y. Q. Gao, H.B.G., Z. W. Qi, et al. , *A multi-objective ant colony system algorithm for virtual machine placement in cloud computing[J].* . Journal of Computer and System Sciences, 2013. 79(8): 1230-1242.

126. L. Wei, C.H.F., B. He, et al., *IEEE Transactions on Cloud Computing, accepted and published online.* Towards efficient resource allocation for heterogeneous workloads in Iaa S clouds [J].

127. Mann., Z.Á., *Multicore-aware virtual machine placement in cloud data centers[J].* IEEE Transactions on Computers, 2016 65(11): 3357-3369.

128. B. Jennings, R.S., *Resource management in clouds: Survey and research challenges[J].* . Journal of Network and Systems Management, 2015. 23(3): 567-619

129. K. Zheng, X.W., L. Li L, et al. , *Joint power optimization of data center network and servers with correlation analysis[C].* in *Proceedings of the IEEE Conference on Computer Communications*. 2014: Toronto.

130. N. Kord, H.H., *An energy-efficient approach for virtual machine placement in cloud based data centers[C].* , in *Proceedings of the 5th Conference on Information and Knowledge Technology*. 2013: Shiraz.

131. X. Wang, X.W., K. Zheng K, et al. , *Correlation-aware traffic consolidation for power optimization of data center networks[J].* IEEE Transactions on Parallel and Distributed Systems, 2016. 27(4): 992-1006

132. K. S. Rao, P.S.T., *Heuristics based server consolidation with residual resource defragmentation in cloud data centers[J].* . Future Generation Computer Systems, 2015 50: 87-98.

133. Crisler, K.J. and M.L. Needham, *Time slot allocation method*. 1997, Google Patents.

134. C. Reiss, J.W., J. L. Hellerstein. Google Inc, *Google cluster-usage traces: format+ schema[R]*, W. Paper, Editor. 2011,.

135. Google cluster-usage traces (version 2), h.c.g.c.p.g., Google Inc., 2014

136. F. Tao, C.L., T. W. Liao, et al. , *BGM-BLA: a new algorithm for dynamic migration of virtual machines in cloud computing[J]*. IEEE Transactions on Services Computing, 2016. 9(6): 910-925

137. M. Dabbagh, B.H., M. Guizani, et al. , *Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment[J]*. IEEE Network, 2015. 29(2): 56-61

139. Gülağız, F.K. and O. Gök. *Estimation of Synchronization Time in Cloud Computing Architecture*. in *International Conference on Mobile Networks and Management*. 2016. Springer.

140. Yu, W., et al., *A survey on the edge computing for the Internet of Things*. IEEE Access, 2018. 6: p. 6900-6919.

141. Yi, S., C. Li, and Q. Li. *A survey of fog computing: concepts, applications and issues*. in *Proceedings of the 2015 workshop on mobile big data*. 2015. ACM.

142. Taneja, M. and A. Davy, *Resource aware placement of data analytics platform in Fog Computing*. Procedia Computer Science, 2016. 97: p. 153-156.

143. Aazam, M. and E.-N. Huh. *Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT*. in *Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on*. 2015. IEEE.

144. Krishnan, Y.N., C.N. Bhagwat, and A.P. Utpat. *Fog computing—Network based cloud computing*. in *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*. 2015. IEEE.

145. Yi, S., Z. Qin, and Q. Li. *Security and privacy issues of fog computing: A survey*. in *International conference on wireless algorithms, systems, and applications*. 2015. Springer.

146. Peng, M., et al., *Fog-computing-based radio access networks: issues and challenges.* Ieee Network, 2016. 30(4): p. 46-53.

147. Gupta, H., et al., *iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments.* Software: Practice and Experience, 2017. 47(9): p. 1275-1296.

148. Gubbi, J., et al., *Internet of Things (IoT): A vision, architectural elements, and future directions.* Future generation computer systems, 2013. 29(7): p. 1645-1660.

149. Wei, P., et al., *Impact Analysis of Temperature and Humidity Conditions on Electrochemical Sensor Response in Ambient Air Quality Monitoring.* Sensors, 2018. 18(2): p. 59.

150. Li, H., Wu, C., Li, Z. and Lau, F.C. *Profit-maximizing virtual machine trading in a federation of selfish clouds.* . in *In INFOCOM, 2013 Proceedings IEEE (pp. 25-29). IEEE*. 2013, April.

151. Bagula, A., D. Djenouri, and E. Karbab. *Ubiquitous sensor network management: The least interference beaconing model*. in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2013.

152. Sehgal, A., *Using the contiki cooja simulator.* Computer Science, Jacobs University Bremen Campus Ring, 2013. 1: p. 28759.

153. Dunkels, A., B. Gronvall, and T. Voigt. *Contiki-a lightweight and flexible operating system for tiny networked sensors*. in *29th annual IEEE international conference on local computer networks*. 2004. IEEE.

154. Hong K., L.D., Ramachandran U. , *Mobile fog：aprogramming model for large-scale applications on the internet of things[C]*, in *ACM SIGCOMM Workshop on Mobile Cloud Computing*. 2013: ACM.

155. Oueis, J., E.C. Strinati, and S. Barbarossa. *The Fog Balancing: Load Distribution for Small Cell Cloud Computing*. in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. 2015.

156. Yangui, S., et al. *A platform as-a-service for hybrid cloud/fog environments*. in *2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 2016.

157. Abedin, S.F., et al. *A Fog based system model for cooperative IoT node pairing using matching theory*. in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2015.

158. Intharawijitr, K., K. Iida, and H. Koga. *Analysis of fog model considering computing and communication latency in 5G cellular networks*. in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*. 2016.

159. Deng, R., et al. *Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing*. in *2015 IEEE International Conference on Communications (ICC)*. 2015.

160. Sarkar, S. and S. Misra *Theoretical modelling of fog computing: a green computing paradigm to support IoT applications*. IET Networks, 2016. 5, 23-29.

161. Ningning, S., et al., *Fog computing dynamic load balancing mechanism based on graph repartitioning*. China Communications, 2016. 13(3): p. 156-164.

162. Ogawa HS, d.O.B., Rodrigues TJ, et al., *Energy consumption and memory footprint evaluation of RPL and CTP in TinyOS In: Proceedings of the XXXIV Simposio Brasileiro de Telecomunicacxoes (SBrT 2016), Santare ́m, PA, 30 August–2 September 2016*. 2016.

163. Felici-Castell S, P.-S.J.J., Segura-Garcia J, et al., *Experimental trade-offs between different strategies for multihop communications evaluated over real deployments of wireless sensor network for environmental monitoring[J].* International Journal of Distributed Sensor Networks, 2018. 14(5): 1550147718774465.

164. K. Machado, D.R., E. Cerqueira, Antonio A. F. Loureiro, A. Neto, and Jos é Neuman de Souza, *“A Routing Protocol Based on Energy and Link Quality for Internet of Things Applications”PMC, PMCID: PMC3649399, February 2013.* .

165. Chiang, M. and T. Zhang, *Fog and IoT: An overview of research opportunities.* IEEE Internet of Things Journal, 2016. 3(6): p. 854-864.

166. Yannuzzi, M., et al. *Key ingredients in an IoT recipe: Fog Computing, Cloud Computing, and more Fog Computing*. in *2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 2014. IEEE.

167. Donassolo, B., et al., *Fog Based Framework for IoT Service Provisioning.* 2018.

168. Aazam, M. and E.-N. Huh. *Fog computing and smart gateway based communication for cloud of things*. in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*. 2014. IEEE.

169. Davis, L., *Handbook of genetic algorithms.* 1991.